Consommation et Corporations Canada
Bureau des brevets
Ottawa, Canada
K1A 0C9

Consumer and Corporate Affairs Canada
Patent Office

(11) (C) **1,314,395**

(21) 564,546

(22) 1988/04/19

(45) 1993/03/16

(52) 35-41

(51) INTL.CL.[5] G09B-19/00

(19)(CA) **CANADIAN PATENT** (12)

(54) Voice Interactive Computer System

(72) Bolin, P. Stanley , U.S.A.
Mason, Reginald B. , U.S.A.

(73) Intechnica International, Inc. , U.S.A.

(30) (US) U.S.A. 040,512 1987/04/20

(57) 5 Claims

564546

Abstract

A voice interactive computer system having voice digitizing circuitry for digitizing voice input from an operator. The voice digitizing circuitry is preferably placed on a computer card in operative association with a central processing unit in the computer. The digitized voice input may be selectively replayed and compared with a prerecorded language vocabulary stored on a compact disc read only memory connected to the computer. The compact disc read only memory is also used for storing software which provides interaction between the voice digitizing circuitry and the computer central processing unit and random access memory in the computer. The voice digitizing circuitry may be placed on a separate computer card positioned in a slot in a bus in operative association with the central processing unit or combined with other cards. The digitized voice input may also be stored on magnetic media such as a computer disc for later review by others, such as a teacher. A method of using the voice interactive computer system in teaching a second language to a student having proficiency in a first language is also disclosed.

VOICE INTERACTIVE COMPUTER SYSTEM

Background Of The Invention

1. Field Of The Invention

This invention relates to voice interactive computer systems, and more particularly, to a computer system usable in a teaching environment and having means for digitizing voice input from a student-operator and for selectively replaying the digitized voice input by the student or a teacher.

2. Description Of The Prior Art

The teaching of foreign languages has traditionally been classroom time intensive. It is necessary to have the interaction between student and teacher so that the student can make the necessary learning connections between speaking, reading and writing. While learning to read the language is important and should not be delayed, multi-sensory input speeds and reinforces the process of acquiring the foreign language, verbally as well as written. In the interaction between the student and teacher, the teacher's proper pronunciation of the foreign language word or phrase is usually repeated as often as necessary, and there is an obvious, immediate aural comparison between the teacher's pronunciation and the student's pronunciation. However, a problem in most classroom situations is that devotion of individual attention by the teacher to a particular student is limited by time constraints. Because of this, the familiar language laboratory has been developed and is in widespread use as an auxiliary teaching tool.

A typical language laboratory utilizes phrases prerecorded in analog form by a language expert. The students can listen to these expert recordings and then repeat the words and phrases. The student's input is recorded in analog form, such as on audio tape media. The student's recorded portion may then be later replayed by the teacher with some systems.

A problem, however, is that there is no easy way for the student to replay what he or she has spoken to compare it with the expert's pronunciation of the same words or phrases. This is true because the student has no real control over the recording equipment. All students hear the expert simultaneously and record their responses simultaneously. Thus, slower students are quickly left behind.

Thus, an important aspect of interaction in language learning is not available in present language laboratories. That is, the student is not able to hear what he or she says and to alternately compare this with the proper pronunciation by the expert. Also, in such language laboratories, the only written materials are preprinted. In other words, there is no immediate correlation between what is spoken and what is written. Again, the student loses an important connection between the verbal and written words or phrases.

The present invention solves these language laboratory deficiencies by providing a voice interactive computer system which allows the student to digitally record his or her spoken words or phrases and immediately replay this recording or the expert prerecording by direct input from the computer keyboard. The student may quickly and easily selectively compare his or her spoken words or phrases with

those pronounced properly by the expert, the expert pronunciation portion being stored in digitized form which may be easily addressed by the computer. Since each student is at a separate computer, each student is in control of his or her learning session.

The student is also presented with a visual display of specific graphics and/or written text at substantially the same time he or she is hearing the verbal counterpart, thus allowing a learning connection between graphic, written and verbal aspects which is not available in language laboratories.

While the system of the present invention cannot totally substitute for individual instruction between a student and teacher, it provides considerably more interaction between expert and student pronunciation and between verbal and written material than does the prior art. Because the student is in control of the system, the student may proceed at his or her own pace.

Summary Of The Invention

The voice interactive computer system of the present invention comprises a computer central processing unit, data entry means in operative association with the central processing unit, memory means in operative association with the central processing unit, software stored in the memory means, and voice digitizing means for digitizing voice input of the operator. The software comprises programming instructions, a digitized voice vocabulary, and graphics and text corresponding to the voice vocabulary. The voice digitizing means also provides a means for retrieving the digi-

tized voice input from memory and replaying the input and also for retrieving at least a portion of the voice vocabulary from the memory means in response to at least one of a direct instruction from the operator and a portion of the programming instructions.

The apparatus further comprises microphone means for receiving the voice input from the operator and transmitting the voice input to the voice digitizing means and speaker means in operative association with the voice digitizing means for audibly reproducing the retrieved voice input and vocabulary portion. Preferably, the microphone means and speaker means are characterized by a headset which frees the hands of the operator.

The voice digitizing means is preferably characterized by a card means positionable in a slot in bus means connected to the central processing unit.

The apparatus further comprises data storage means in operative association with the central processing unit. Preferably, the voice digitizing means also comprises means for storing the digitized voice input in the data storage means. The data storage means comprises at least one of another memory means and a disc storage means. The other memory means may be characterized by a random access memory.

In the preferred embodiment, the first mentioned memory means comprises a compact disc read only memory.

One preferred method of using the computer system of the present invention is for teaching a second language to a student having some proficiency in a first language. The method comprises the steps of storing a voice language vocabulary in digitized form in memory means in operative asso-

ciation with a computer central processing unit, placing voice digitizing means in operative association with the central processing unit, storing software in the memory means and running the software in the central processing unit for providing interaction between the central processing unit and the voice digitizing means, digitizing an analog voice input signal from the student, storing the digitized voice input in the memory means, and comparing the digitized voice input with at least a portion of the language vocabulary. Graphics and text corresponding to the voice vocabulary may also be stored in the memory means.

The step of storing the vocabulary preferably comprises prerecording the vocabulary on a compact disc read only memory, and the step of storing the software comprises placing the software on the compact disc read only memory. The graphics and text are also prerecorded on the compact disc read only memory.

The step of placing the voice digitizing means in operative association with the central processing unit comprises placing voice digitizing circuitry on a computer card and positioning the card in the computer.

The step of storing the digitized voice input comprises storing the voice input in a random access memory in operative association with the central processing unit of the computer.

The step of comparing the digitized voice input with the portion of the vocabulary comprises selectively retrieving the voice input or vocabulary portion from the memory means and selectively replaying the digitized voice input and the portion of the vocabulary in analog form. Graphics and text

may be shown on monitor means substantially simultaneously for intimate interaction with the audio.

The method of teaching a foreign language further comprises the step of storing the voice input in storage means for subsequent review by the teacher of the student. This step of storing the voice input preferably comprises storing the input on magnetic media, such as a computer disc.

It is an important object of the present to provide a computer system having means for digitizing voice input from an operator and for selectively replaying the digitized voice input.

It is another object of the invention to provide a computer system with intimately interactive audio, graphics and text.

It is a further object of the invention to provide a computer system suitable for teaching a second language to a student having proficiency in a first language.

It is an additional object of the invention to provide a computer system having voice digitizing means for digitizing voice input of an operator and retrieving the digitized voice input and for also retrieving at least a portion of a voice vocabulary stored in memory means, such retrieval being in response to a direct instruction from an operator or from software programming instructions.

A further object of the invention is to provide a language teaching computer system having a voice vocabulary stored on a compact disc read only memory.

Still another object of the invention is to provide a method of teaching a second language to a student having

some proficiency in a first language in which digitized voice input of the student may be compared with a portion of a prerecorded language vocabulary.

Additional objects and advantages of the invention will become apparent as the following detailed description of the preferred embodiments is read in conjunction with the drawings which illustrate such preferred embodiments.

Brief Description Of The Drawings

FIG. 1 shows a schematic of a preferred embodiment of the voice interactive computer system of the present invention.

FIG. 2 shows a schematic of an alternate embodiment of the invention.

FIG. 3 is a functional block diagram of the voice card used in the invention.

FIGS. 4A and 4B are together a circuit schematic showing the analog circuitry in the voice card.

FIG. 5 shows the relationship between FIGS. 5A, 5B, 5C and 5D which are together a circuit schematic of the digital circuitry of the voice card.

FIGS. 6-19 illustrate flow charts for the computer software main program used in the system.

Description Of The Preferred Embodiment

Referring now to the drawings, and more particularly to FIG. 1, an embodiment of the voice interactive computer system of the present invention is shown and generally designated by the numeral 10. The major hardware components of system 10 include a computer 12 and a module 14, although as will be understood herein, computer 12 and module 14 could be combined into a single housing if desired.

Computer 12 is of a kind generally known in the art, such as an IBM PC XT*, although the system may be adapted to virtually any kind of computer and is not intended to be limited to an IBM PC*. Computer 12 includes data entry means such as a keyboard 16 and data storage means, such as a disc drive 18.

Disc drive 18 may include one or more of any known disc drives, such as the various sizes of floppy disc drives or hard disc drives. The invention is not intended to be limited to any particular data storage means.

Both the keyboard and disc drive are in operative association with a central processing unit (CPU) 20 of computer 12 and thus also in operative association with a memory means 22 such as a random access memory (RAM), all of a kind known in the art.

Forming a portion of computer 12 and connected to the other components thereof is a row of computer card slots or slot bus 24, also called an input/output bus or I/O bus. Again, slot bus 24 is of a kind known in the art and is adapted for receiving various generally known computer cards. Slot bus 24 includes a plurality of slots such as 26, 28, 30, 32 and 34. The number of slots is not critical and is not intended to be a limiting feature of the invention.

Also connected to central processing unit 20 and forming a portion of computer 12 is a cathode ray tube (CRT) monitor 36 on which is displayed data in a normal manner. All of the components of computer 12 may be enclosed within a single housing 38 or separate housings for the various components as desired.

* trade-marks.

Module 14 preferably includes a housing 40 which encloses a row of card slots or slot bus 42 similar to slot bus 24 in computer 12. In FIG. 1, slot bus 42 includes slots 44, 46, 48, 50 and 52, but the number is not critical.

Another component included in module 14 is another memory means preferably in the form of a compact disc read only memory (CDROM) drive 54. CDROM drive 54 is of a kind known in the art and is preferred because of the large amount of memory storage available thereon. However, any other type of memory storage means such as read only memory (ROM) chips or a hard disc drive may be utilized as long as the memory capacity thereof is sufficient. The invention is not intended to be limited to a CDROM drive.

CDROM drive 54 is connected by a cable 56 to a CDROM interface card 58. Interface card 58 is in turn connected to slot bus 42 in module 14 by plugging interface card 58 into one of the slots, for example slot 52.

A bus interface 60 interconnects slot bus 24 in computer 12 with slot bus 42 in module 14. Bus interface 60 includes a cable 62 having a first bus interface card 64 at one end of the cable and a second bus interface card 66 at the other end of the cable. First bus interface card 64 plugs into one of the slots, such as slot 28, in slot bus 24 in computer 12. Second bus interface card 66 plugs into one of the slots, such as slot 44, in slot bus 42 in module 14. Bus interface 60 may be of any kind known in the art and is generally referred to as a transmitter card.

It is contemplated that a plurality of computers may be connected to module 14. For example, a second computer 12', substantially identical to first computer 12, may be con-

nected to module 14 by plugging second bus interface card 66' thereof into another slot, such as slot 46, in slot bus 42 in the module. Additional computers may also be connected to module 14 in a similar manner. The only physical limitation is the number of slots available in slot bus 42.

Computer system 10 also comprises voice digitizing means such as a voice card 68 plugged into another slot in slot bus 24 in computer 12, such as slot 26. Other computers in system 10 also have such voice digitizing means plugged thereinto. Connected to voice card 68 by a cable 70 are speaker and microphone means. Preferably, this is in the form of a headset 72 in which the speaker means includes an earphone 74, and the microphone means is a microphone 76 attached thereto. A headpiece 78 allows the operator of computer 12 to wear headset 72 while keeping his or her hands free to operate the computer. A separate speaker and microphone could also be utilized if desired.

Using control through keyboard 16, voice card 68 receives voice input from the operator through microphone 76 and digitizes the voice input in a manner hereinafter described in more detail. An optional manual control switch 79 may be placed in cable 70 to control microphone 76 externally of keyboard 16 as desired.

Referring now to FIG. 2, an alternate embodiment of the voice interactive computer system of the present invention is shown and generally designated by the numeral 80. System 80 also includes a computer 12 and a module 14. As with the first embodiment, computer 12 in system 80 includes a housing 38 which houses a keyboard 16, data storage means, such as disc drive 18, a central processing unit 20, memory

means, such as random access memory 22, a slot bus 24 and a monitor 36. Again, separate housing portions for the various components may be utilized as desired.

Module 14 in alternate system 80 again includes another memory means, such as a CDROM drive 54 and a slot bus 42.

A combination bus interface/voice card/CDROM interface, generally designated by the numeral 82, interconnects computer 12 and module 14 in system 80. Interface 82 includes a cable 84 having a first interface card 86 at one end thereof which is plugged into a slot, such as slot 26, in slot bus 24 in computer 12. Interface card 86 includes all of the components and performs the same functions as first bus interface card 64 and voice card 68 in first system 10. By combining the circuit components on a single card, the amount of hardware is reduced which decreases the cost. An additional advantage is that another slot, such as slot 28, is freed in computer 12 for other usage.

The other end of cable 84 is connected to a second interface card 88 which plugs into a slot, such as slot 44, in slot bus 42 in module 14. Second interface card 88 includes all of the components and performs the same functions as CDROM interface card 58 and first bus interface card 66 in first system 10. Again, a card is eliminated which reduces the cost, and an additional slot, such as slot 52, is freed in module 14 for other usage.

Headset 72 is connected to interface card 86 by a cable 90, and CDROM drive 54 is connected to second interface card 88 by a cable 92. Optional switch 79 may be placed in cable 90.

As with first embodiment 10, it is contemplated that multiple computers, such as computer 12', may be utilized in

system 80. Computer 12' is connected to module 14 by interface 82' having a cable 84' with a second interface card 88' plugged into a slot, such as slot 46, in slot bus 42 in the module. A cable 92' interconnects second interface card 88' with CDROM drive 54. Additional computers may be connected in a similar manner.

Referring now to FIGS. 3-5, the details of voice card 68 of system 10 or the voice card portion of interface card 86 of the system 80 will be described. FIG. 3 is a functional block diagram of the circuitry schematics shown in FIGS. 4A and B and 5A-D. FIGS. 4A and B show the analog circuitry required for filtering and amplification, and FIGS. 5A-D show the digital circuitry required to interface to slot bus 24 in computer 12 as well as to convert the analog voice signal to digital information. Reference will be made in this discussion only to voice card 68 of system 10, but it should be understood that this applies equally to the voice card portion of interface card 86 of alternate system 80.

In addition to the reference numerals herein, specific electrical components shown in the circuit schematics will be identified by the reference codes shown in those schematics.

In block 94, the audio signal from microphone 76 must be amplified to a suitable level by integrated circuit 96 (IC8) to interface through line 98 to analog to digital converter 100 (IC14). The signal must pass through one-half of low pass filter 102 (IC11) before being converted to a digital signal. Low pass filter 102 will attenuate any voice frequencies which are one-half of a sampling frequency.

In analog to digital converter 100, the analog signal is momentarily saved in a sample and hold circuit. Converter

100 is an eight bit successive capacitor ladder conversion. The output of analog digital converter 100 is then sent as a parallel eight bit word to parallel to serial shift register 104 (IC13) through lines 106.

Shift register 104 accepts a parallel signal from analog to digital converter 100, and the shift register can then be clocked to allow a serial stream of data to be fed to an adaptive delta pulse code modulation (ADPCM) processing chip 108 (IC12). Processing chip 108 performs the ADPCM algorithm on the incoming data and places the results on lines 110 which are connected to slot bus 24 of computer 12 through standard interface chips 112 (IC21), 114 (IC22) and 116 (IC23). Chip 112 serves as an input buffer, and chip 114 serves as an output buffer. Input/output lines 118 (D0-D7) directly connect to slot bus 24, and chips 112, 114 and 116 allow processing chip 108 to talk to slot bus 24. Chips 112, 114 and 116 are bi-directional and tristateable. Thus, processor chip 108 may both send data to computer 12 and accept data from the computer.

Timing control is provided by timing control circuit 120 which is connected to clock circuit 122. Clock circuit 122 includes an oscillator 124 (OSC1) and a counter 126 (IC15) known in the art.

In block 128, previously digitized data may be read from memory 22 of computer 12 and input to the data bus of processing chip 108. Processing chip 108 converts the digital information back to an analog signal. It is necessary to both amplify and filter the signal so that the original speech signal will sound suitable to the operator. This amplification is done by integrated circuit 130 (IC10), and

the filtering is accomplished by the other half of low pass filter 102 (IC11).

The audio signal is amplified by output amplifier 132 (IC9) to a level sufficient to drive earphone 74 of headset 72 or a separate speaker.

In block 134, integrated circuits 136 (IC18), 138 (IC19), and 140 (IC20) allow computer 12 to input to the circuitry through address lines 142 (AD4-AD9 and -AEN), write line 144 (-IOW) and read line 146 (-IOR).

Switches 148 (SW2) can be set to decode a unique address in connection with the signals through write line 144 and read line 146 which will ultimately control the operation of processing chip 108, and thus the voice functions themselves.

It will thus be seen that the complete circuit of voice card 68 is capable of digitizing an analog voice signal and storing the information in the memory of computer 12. Likewise, previously recorded digital information can be converted back to an analog voice signal. All of this, of course, is under the control of software programs written to support this circuitry.

In the preferred embodiment, the software includes a computer program written in a high level language marketed under the trade-mark "Tencore". Tencore is a commercially available language produced by Computer Teaching Corporation, of Champaign, Illinois. A listing of the program is included as Appendix A in the specification. In addition, the software includes an assembly language program which ties voice card 68 or the voice card portion of interface card 86 into the Tencore language program. A listing of the assembly language program is in Appendix B in the specification.

FIG. 6 is a flow diagram of the overall Tencore language program. A flow chart of the Introduction portion of the program is shown in FIG. 7. FIG. 8 presents a detailed flow chart of the Menu Scanner Routine indicated in FIG. 7.

The main program includes portions for numbers, vocabulary and grammar. The Main Numbers Program is described in the flow chart shown in FIG. 9. The flow chart of the Numbers Instruction Program is shown in FIG. 10, and a flow chart of the Numbers Instruction Routine is given in FIG. 11. FIG. 12 shows a flow chart of the Numbers Drill Routine.

A flow chart of the Vocabulary Instruction Portion of the main program is shown in FIG. 13. The Vocabulary Learn and Review portion is shown in the flow chart of FIG. 14, and the Vocabulary Drill is illustrated in the flow chart of FIG. 15.

The flow chart in FIG. 16 shows the Grammar Drill portion.

FIG. 17 gives a flow chart of the Text Play and Repeat portion of the main program, and FIG. 18 is a flow chart of the Repeat Text Routine.

The Dictation portion of the program is presented in the flow chart of FIG. 19.

The Tencore language program of Appendix A and the assembly language program of Appendix B both have detailed comments printed in the program listings. A person skilled in the art will easily understand the software used in the present invention after a study of the flow charts of FIGS. 6-19 and the program listings in Appendices A and B.

The software is preferably stored in CDROM drive 54 and is activated upon start-up of either system 10 or 80. When

the software is running in computer 12, the various functions of voice card 68 or the voice card portion of interface card 86 are activated. Again, reference in this discussion is made only to voice card 68, but this applies also to the voice card portion of interface card 86.

In using the system in a language-teaching situation, the operator or student is not required to be a computer expert. Basically, the system is turned on, and the software then functions and instructs the student throughout the process.

For teaching foreign languages, an extensive voice vocabulary is stored in CDROM drive 54 including the native or first language of the student as well as the language being taught. In most cases, one of these languages will be English. However, the invention is not intended to be limited to the teaching of foreign languages to English-speaking students. It applies equally well, and perhaps with more social impact, to the teaching of English as a second language to non-English-speaking persons. This is a particularly important function sociologically so that non-English-speaking persons can be more easily assimilated into, and work within, an English-speaking society.

Once the system is running and the student is wearing head set 72, the student may speak into microphone 76 when instructed by the program. Control may be taken through keyboard 16 or optional switch 79 for recording this voice input is digitized by voice card 68 and stored in memory in computer 12, ordinarily RAM 22. For example, the computer may display a foreign language phrase and the English equivalent thereof in text form on monitor 36 as well as

giving an aural signal to the student of the phrase through earphone 74 of headset 72. Corresponding graphics may also be displayed on monitor 36. The student may then verbally repeat the foreign language word or phrase which is then stored in the system as described. By simple key strokes, on keyboard 16, the student may then replay the professionally spoken phrase from the voice vocabulary in the system and also replay his or her own spoken version of the word or phrase. Both can be replayed as many times as desired so that the student gets a true interaction with the system. The student may make additional attempts to properly pronounce the word or phrase as the student desires. Because the student is in control of the system, the student may proceed at his or her own pace.

The software and vocabulary may be written to provide any number of verbal and written exercises as desired. In all cases, the student may immediately replay his or her spoken version of the phrase and compare it to the proper pronunciation. By such repetition of this verbal aspect, there is increased comprehension.

Upon start-up of the system, the student will place memory media, such as a floppy disc, in disc drive 18, and the system will automatically record his or her spoken words or phrases on the student's disc. This data disc may then be reviewed at a later time by a teacher for evaluation and additional instruction as necessary.

Because of the unique digitized recording of the student's voice and ability to replay on command, along with corresponding graphics and text, the system provides an interaction much closer to that of a teacher-student

classroom interaction than with previously known devices such as language laboratories.

While the system has been described in particular for a language-teaching situation, it will be seen that by modification of the software, the system is easily adaptable for other voice interactive usages.

It can be seen, therefore, that the voice interactive computer system of the present invention is well adapted to carry out the ends and advantages mentioned, as well as those inherent therein. While presently preferred embodiments of the invention have been described for the purposes of this disclosure, numerous changes in the arrangement and construction of parts may be made by those skilled in the art. All such changes are encompassed within the scope and spirit of the appended claims.

1314395

APPENDIX A

```
* This defines file has been tailored for listing comprehension strand.

rcount,2                $$ number of right answers
wcount,2                $$ number of wrong answers
attempts, 2             $$ total number of attempts

TRUE = -1
FALSE = 0

ON = -1
OFF = 0

debug = FALSE
talk = TRUE

type,1                  $$ type of picture currently being displayed
pcx = 0
pcc = 1

offsety = 0     $$ number to be added to zpy from light pen
offsetx = 0     $$ number to be added to zpx from light pen

lpy,2           $$ zpy + offsety
lpx,2           $$ zpx + offsetx

pictnam,8       $$ current picture generic name

* TENSPEAK command list

teninit = 0     $$ unit TENSPEAK to known state
topen = 1       $$ open speech archive filespec.tfl
tsay = 2        $$ speak entry n of speech archive previously opened
tclose = 3      $$ close previously opened speech archive
tread = 4       $$ read entry n into speech buffer only
tspeak = 5      $$ speak previously loaded speech
tload = 6       $$ speak filename.rec
trecord = 7     $$ record speech into memory
```

```
38  tsave = 8       $$ save memory buffer on disk
39
40  tenspeak,8      $$ tenspeak interrupt call register list; ax, bx, cx, dx
41  •       ah,1    $$ entry number (is required) is passed in here
42  •       al,1    $$ tenspeak command (0 - 6) is passed from here
43
44  •       bx,2    $$ offset to talk archive name is passed here
45  •       ch,1
46  •       cl,1    $$ number of entries returned here in topen
47  filespec,45     $$ talk archive filespec
48  * {drive:}\direct1\direct2\direct3\filename{.ext},[space,0,CR]
49
50  * picture handler defines
51  picture = 9
52  plot = 1
53  read = 2
54  replot = 3
55  release = 4
56
57  boxkey, 1       $$ box last selected by the lightpen
58  pnone    = 0    $$ none of the five boxes was chosen
59  pcont    = 1    $$ continue (SIGA)
60  ppause   = 2    $$ pause (PAUSA)
61  prepeat  = 3    $$ repeat (REPITA)
62  phelp    = 4    $$ help (AYUDE)
63  plesson  = 5    $$ lesson (LECCION)
64
65  balloonx = 00   $$ x,y location of talk balloon
66  balloony = 349
67  cluex    = 280
68  cluey    = 349
69  questx   = 16
70  questy   = 184
71  answerx  = 280
72  answery  = 184
73  inputx   = 16
74  inputy   = questy-3*28
75  graphx   = 66
76  graphxw  = 570      $$ 8.5"
```

```
graphy   = 184
graphyw  = 124      $$ 2.4"

field,1             $$ number of field to erase
* erasefld field values
balloon  = 0
clue     = 1
quest    = 2
answer   = 3
input    = 4
half     = 5
graph    = 6

* defines for bbx locations
uppery   = 55
lowery   = 25
texty    = 32
box1     = 87
box2     = 187
box3     = 287
box4     = 387
box5     = 487

* HELP menu equates
helpkey,1           $$ current help request

hlpintro = 1
hlphowky = 2
hlpalpha = 3
hlpalflt = 4
hlpalfwd = 5
hlpnumbr = 6
hlpnumwd = 7
hlpsymbl = 8
hlpwngam = 9
hlpmoney = 10
hlptelep = 11
hlpnames = 12
hlpgreet = 13
```

```
116  hlpthth  = 14
117  hlppast  = 15
118  helpgram = 16
119  hipvocab = 17
```

```
120  initial startup
121  spacing variable
122  screen  native
123  thick   on
124  options current
125
126  ***** Initialization *****
127
128  do      tenload          $$ check to see if TENSPEAK is loaded
129
130  do      speak(teninit)   $$ make sure it is initialized
131
132  ***** SCREEN 1 *****
133
134  colore  cyan+
135  erase   0,349;639,199
136
137  calc    pictnam == 'world'
138  do      picture(pcc,000,32,335)     $$ picture of world
139
140  calc    pictnam == 'logo'
141  do      picture(pcc,000,164,300)    $$ Intechnica Logo
142
143  calc    pictnam == 'entut'
144  do      picture(pcc,000,100,185)    $$ Student Course
145                                      $$ Introduction in
146                                      $$ English
147  at      24:40
148  size    2
149  color   red+
150  write   Hit star to continue
151
152  do      screen(001)
153
154  pause   Keys=next
155
156  colore  black
```

```
157
158  erase
159
160  zero     filespec
161  pack     filespec,,\tesol\spint\entutor1
162  do       speak(topen)      $$ entutor1.tf1
163
164  calc     pictnam == 'spint'
165  do       picture(pcc,000,balloonx,balloony)   $$ talk balloon
166
167  do       boxes
168
169  ***** SCREEN 2 *****
170
171  size     2
172  color    blue
173  at       balloonx+48,balloony-57
174  write    Hello!
175           My name is
176  VoxBox.
177
178  do       speak(tsay,000)        $$ "Hello!"
179  do       speak(tsay,001)        $$ "My name is VoxBox."
180
181  do       screen (002)
182
183  do       debug
184
185  ***** SCREEN 3 *****
186
187  do       erasefld(balloon)
188
189  size     2
190  at       balloonx+32,balloony-43
191  write    You and I
192           are beginning
193           an exciting
194           adventure,
195
```

```
196  do      speak(tsay,002)        $$ "You and I are beginning an exciting
197                                 $$ adventure."
198
199  do      screen (003)
200
201  do      debug
202
203  ***** SCREEN 4 *****
204
205  do      erasefld(balloon)
206
207  at      balloonx+48,balloony-57
208  size    2
209  write   We are going
210          to learn
211          English.
212
213  do      speak(tsay,003)        $$ "We are going to learn English."
214
215  do      screen(004)
216
217  do      debug
218
219  ***** SCREEN 5 *****
220
221  do      erasefld(balloon)
222
223  do      speak(tsay,004)
224
225
226  do      picture(pcc,001,graphx+66,graphy-25)
227
228  do      speak(tsay,005)        $$ "ENGLISH."
229
230  do      speak(tsay,006)        $$ "see the word on the screen?  Say
231                                 $$ after me."
232  do      speak(tsay,007)        $$ "English..."
233
234  delay   1
```

```
235
236  do      speak(tsay,008)        $$ "ENGLISH"
237
238  do      speak(tsay,009)        $$ "once more"
239
240  do      speak(tsay,010)        $$ "English"
241
242  delay   1
243
244  do      speak(tsay,011)        $$ "ENGLISH"
245
246  do      speak(tsay,012)        $$ "Very good."
247
248  do      speak(tsay,013)        $$ "from now on I will say English and I
249                                     hope you will say English."
250  do      screen(005)
251
252  do      debug
253
254  ***** SCREEN 6 *****
255
256  do      erasefld(balloon)
256
258  do      speak(tsay,014)        $$ "When we have completed all the lessons"
259
260  at      balloonx+48,balloony-43
261  size    2
262  write   You will be
263          able to...
264
265  do      speak(tsay,015)        $$ "You will be able to..."
266
267  do      screen(006)
268
269  do      debug
270
271  ***** SCREEN 7 *****
272
273  at      balloonx+48,balloony-99
```

```
size    2
write   speak English

do      picture (pcc,002,graphx,graphy)      $$ mouth speaking English

do      speak(tsay,016)       $$ "speak ENGLISH"

do      screen(007)

do      debug

***** SCREEN 8 *****

do      erasefld(graph)
do      erasefld(half)

at      balloonx+32,balloony-99
size    2
write   understand
        spoken English

do      picture(pcc,003,graphx+16,graphy)    $$ not hearing English.
* NOTE:  had to add +16 to graphx because spint003,pcc was leaving a cyan
* line to the left of the graphx margin.

do      speak(tsay,017)       $$ "understand spoken English"

do      screen(008)

do      debug

***** SCREEN 9 *****

do      erasefld(graph)
do      erasefld(half)

at      balloonx+48,balloony-99
size    2
write   read and
```

```
313           write English
314
315   do      picture(pcc,004,graphx+16,graphy)    $$ book and English and pencil
316
317   do      speak(tsay,018)        $$ "read English and write English"
318
319   do      screen(009)
320
321   do      debug
322
323   ***** SCREEN 10 *****
324
325   do      erasefld(graph)
326   do      erasefld(balloon)
327
328   at      balloonx+48,balloony-57
329   size    2
330   write   This is how
331           we will learn
332           together.
333
334   do      speak(tsay,019)        $$ "How will we do all this?  This is how
335                                  $$ "we will learn together."
336
337   do      screen(010)
338
339   do      debug
340
341   ***** SCREEN 11 *****
342
343   do      erasefld(balloon)
344
345   at      balloonx+48,balloony-57
346   size    2
347   write   I will talk to
348           you through
349           the headset.
350
351   do      picture(pcc,005,cluex+60,cluey)      $$ picture of the headset
```

```
352  do      picture(pcc,999,graphx+16,graphy-35)      $$ headset text
353
354  do      speak(tsay,020)          $$ "I will talk to you through the headset
355                                   $$ "you are now wearing
356
357  do      speak(tsay,021)          $$ "the English word for headset is..."
358
359  do      speak(tsay,022)          $$ "headset"
360
361  do      speak(tsay,023)          $$ "say headset"
362
363  do      speak(tsay,024)          $$ "you are now wearing your headset"
364
365  do      screen(011)
366
367  do      debug
368
369  ***** SCREEN 12 *****
370
371  do      erasefld(clue)
372  do      erasefld(graph)
373  do      erasefld(balloon)
374
375  at      balloonx+32,balloony-57
376  size    2
377  write   You will talk
378          to me through
379          the Keyboard.
380
381  do      picture(pcc,006,graphx+16,graphy)    $$ Keyboard picture
382
383  do      speak(tsay,025)          $$ "you will talk to me through the
384                                   $$ "Keyboard."
385
386  do      screen (012)
387
388  do      debug
389
390  ***** end of page 1 *****
```

```
391 jumpop page2
392
```

```
screen  native
spacing variable
thick   on

if      debug=TRUE
•       calc    pictnam == 'spint'
•       do      picture(pcc,000,balloonx,balloony)
•       do      boxes
•       do      speak(teninit)
•       zero    filespec
•       pack    filespec,,\tesol\spint\entutor1
•       do      speak(topen)
endif

***** SCREEN 13 *****

do      erasefld(graph)
do      erasefld(balloon)

size    2
color   blue
at      balloonx+48,balloony-57
write   ... and with
        your
        Lightpen.

do      picture(pcc,007,graphx,graphy)        $$ picture of lightpen.

do      speak(tsay,026)         $$ "and with your lightpen. This is what it
                                $$ looks like... your lightpen is on the
                                $$ desk in front of you."
do      speak(tsay,027)         $$ "Please pick it up."

do      screen (013)

do      debug
```

```
430  ***** SCREEN 14 *****
431
432  do      erasefld(balloon)
433
434  size    2
435  at      balloonx+32,balloony-57
436  write   You tell me
437          what to do
438          by touching.
439
440  do      speak(tsay,028)          $$ "With your lightpen you are always in
441                                   $$ control because you tell me what to do
442                                   $$ by touching the lightpen to the light-
443                                   $$ boxes now on the screen."
444  do      speak(tsay,029)          $$ "and since we will use the lightpen and
445                                   $$ lightboxes so much..."
446  do      speak(tsay,030)          $$ "I will now teach you these words in
447                                   $$ "English."
448
449  do      screen(014)
450
451  do      debug
452
453  ***** SCREEN 15 *****
454
455  do      erasefld (balloon)
456  do      erasefld(graph)
457
458  at      balloonx+32,balloony-57
459  size    2
460  write   The English
461          word for
462          Lightpen is...
463
464  do      picture(pcc,899,graphx+85,grahy-25)       $$ text "lightpen"
465
466  do      speak(tsay,031)          $$ "The English word for lightpen is
467                                   $$ lightpen
468
```

```
469  do      speak(tsay,032)          $$ "lightpen"
470
471  delay   1
472
473  do      speak(tsay,033)          $$ "LIGHTPEN"
473
475  do      speak(tsay,034)          $$ "say after me.."
476
477  do      speak(tsay,035)          $$ "lightpen"
478
479  delay   1
480
481  do      speak(tsay,036)          $$ "LIGHTPEN"
482
483  do      speak(tsay,037)          $$ "once more"
484
485  do      speak(tsay,038)          $$ "lightpen"
486
487  delay   1
488
489  do      speak(tsay,039)          $$ "LIGHTPEN"
490
491  do      speak(tsay,040)          $$ "excellent!"
492
493  do      screen(015)
494
495  do      debug
496
497  ***** SCREEN 16 *****
498
499  do      erasefld(graph)
500  do      erasefld(balloon)
501
502  do      speak(tsay,041)          $$ "Now..."
503
504  at      balloonx+32,balloony-57
505  size    2
506  write   The English
507          word for
```

```
508           Lightbox is...
509
510   do      picture(pcc,008,graphx+85,graphy-25)      $$ lightbox text
511
512   do      speak(tsay,042)         $$ "the English word for lightbox is
513                                     $$ Lightbox"
514   delay   1
515
516   do      speak(tsay,043)         $$ "lightbox"
517
518   delay 1
519
520   do      speak(tsay,044)         $$ "LIGHTBOX"
521
522   do      speak(tsay,045)         $$ "say after me.."
523
524   do      speak(tsay,046)         $$ "lightbox"
525
526   delay   1
527
528   do      speak(tsay,047)         $$ "LIGHTBOX"
529
530   do      speak(tsay,048)         $$ "once more"
531
532   do      speak(tsay,049)         $$ "lightbox"
533
534   delay   1
535
536   do      speak(tsay,050)         $$ "LIGHTBOX"
537
538   do      speak(tsay,051)         $$ "you say it very well"
539
540   do      screen(016)
541
542   do      screen(016)
543
544   do      debug
545
546   ***** SCREEN 17 *****
```

```
547
548  do      erasefld(graph)
549  do      erasefld(balloon)
550
551  at      balloonx+48,balloony-43
552  size    2
553  write   Use your
554          Lightpen to
555          touch the
556          Lightbox.
557
558  do      speak(tsay,052)          $$ "from now on I will tell you to use
559                                   $$ your lightpen to touch the lightbox."
560
561  do      screen(017)
562
563  do      debug
564
565  ***** SCREEN 18 *****
566
567  do      erasefld(balloon)
568
569  do      speak(tsay,053)          $$ "Now..."
570
571  at      balloonx+24,balloony-57
572  size    2
573  write   when you touch
574          your Lightpen
575          to a Lightbox.
576
577  do      lightpen(320,100,ON)
578
579  do      speak(tsay,054)          $$ "we will learn what happens when you
580                                   $$ touch your lightpen to a lightbox"
581
582  do      screen(018)
583
584  do      debug
585
```

```
586  ***** SCREEN 19 *****
587
588  do      erasefld(balloon)
589  do      lightpen(320,100,OFF)
590
591  do      speak(tsay,055)          $$ "Across the bottom of the screen"
592
593  do      speak(tsay,056)          $$ "starting on the left..."
594
595  do      speak(tsay,057)          $$ "the first word..."
596
597  do      speak(tsay,058)          $$ "CONTINUE"
598
599  do      flipbox(pcont)           $$ highlight the continue box
600
601  do      screen(019)
602
603  do      debug
604
605  ***** SCREEN 20 *****
606
607  at      balloonx+40,balloony-43
608  size    2
609  write   CONTINUE
610          means that
611          when you are
612          ready...
613
614  do      speak(tsay,059)          $$ "means that when you are ready..."
615
616  do      speak(tsay,060)          $$ "and want to go to the next lesson."
617
618  do      boxes(pcont)             $$ put continue back for next demo
619
620  do      speak(tsay,061)          $$ "I will do so when you touch your
621                                   $$ lightpen to the CONTINUE word."
622
623  do      lightpen(box1+33,upper,ON)      $$ touch lightpen to continue
624
```

```
625  do      flipbox(pcont)          $$ highlight the box
626
627  delay   2
628
629  do      screen(020)
630
631  do      debug
632
633  do      lightpen(box1+33,uppery,OFF)
634
635  do      boxes(pcont)            $$ put back continue original
636
637  ***** end of page 2 *****
638
639  jumpop page3
```

```
640  screen  native
641  spacing variable
642  thick   on
643
644  if      debug=TRUE
645  •       calc    pictnam == 'spint'
646  •       do      picture(pcc,000,balloonx,balloony)
647  •       do      boxes
648  •       do      speak(teninit)
649  •       zero    filespec
650  •       pack    filespec,,\tesol\spint\entutor1
651  •       do      speak(topen)
652  endif
653
654  ***** SCREEN 21 *****
655
656  do      erasefld(balloon)
657
658  at      balloonx+32,balloony-43
659  color   blue
660  size    2
661  write   PAUSE simply
662          tells me to
663          wait until you
664          are ready,
665
666  do      lightpen(box2+33,uppery+1,ON)        $$ touch lightpen to pause
667
668  do      flipbox(ppause)          $$ highlight the box
669
670  do      speak(tsay,062)          $$ "The word PAUSE simply tells me to
671                                   $$ wait until you are ready to go on
672                                   $$ with the lesson you are working on"
673
674  do      screen(021)
675
676  do      debug
```

```
677
678  do      lightpen(box2+33,uppery+1,OFF)
679
680  do      boxes(ppause)
681
682  ***** SCREEN 22 *****
683
684  do      erasefld(balloon)
685
686  at      balloonx+16,balloony-43
687  size    2
688  color   blue
689  write   REPEAT lets
690          you go back and
691          do the last
692          exercise.
693
694  *do     speak(tsay,063)          $$ "The third word..."
695
696  do      speak(tsay,064)          $$ "REPEAT"
697
698  do      lightpen(box3+33,upper+1,ON)          $$ touch lightpen to repeat
699
700  do      flipbox(prepeat)         $$ highlight the box
701
702  do      speak(tsay,065)          $$ "lets you go back and do the last exercise
703                                   $$ you just worked on."
704
705  do      speak(tsay,066)          $$ "as often as you like."
706
707  do      screen(022)
708
709  do      debug
710
711  do      lightpen(box3+33,upper+1,OFF)
712
713  do      boxes(prepeat)
714
715  ***** SCREEN 23 *****
```

```
716
717  do      erasefld(balloon)
718
719  at      balloonx+16,balloony-43
720  size    2
721  color   blue
722  write   HELP means I am
723          always ready
724          to assist when
725          trouble occurs.
726
727  do      speak(tsay,067)        $$ "The Lightbox word..."
728
729  do      speak(tsay,068)        $$ "HELP"
730
731  do      lightpen(box4+33,upper+1,ON)          $$ touch lightpen to help
732
733  do      flipbox(phelp)         $$ highlight the box
734
735  do      speak(tsay,069)        $$ "means I am always available to come
736                                 $$ running when you run into trouble."
737
738  do      screen(023)
739
740  do      debug
741
742  do      lightpen(box4+33,upper+1,OFF)
743
744  do      boxes(phelp)
745
746  ***** SCREEN 24 *****
747
748  do      erasefld(balloon)
749
750  at      balloonx+16,balloony-43
751  size    2
752  color   blue
753  write   When you touch
754          HELP with your
```

```
755             Lightpen, watch
756             what happens.
757
758   do        speak(tsay,070)         $$ "When you touch HELP with your Lightpen
759                                       $$ I will show you all the ways I can help
760                                       $$ you..."
761   do        speak(tsay,071)         $$ "WATCH!"
762
763   do        lightpen(box4+33,upper+1,ON)          $$ touch lightpen to help
764
765   do        flipbox(phelp)
766
767   do        screen(024)
768
769   do        debug
770
771   do        speak(tclose)           $$ close entutor1.tf1
772
773   ***** SCREEN 25 *****
774
775   zero      filespec
776   pack      filespec,,\tesol\spint\entutor2
777   do        speak(topen)            $$ open entutor2.tf1
778
779   do        helpmenu(65,295)
780
781   do        screen(025)
782
783   do        speak(tsay,000)         $$ "I will help you with:"
784   do        speak(tsay,001)         $$ "Introduction"
785   do        speak(tsay,002)         $$ "How to use your keyboard"
786   do        speak(tsay,003)         $$ "Alphabet"
787   do        speak(tsay,004)         $$ "Letters"
788   do        speak(tsay,005)         $$ "Words"
789   do        speak(tsay,006)         $$ "Numbers"
790   do        speak(tsay,007)         $$ "Words"
791   do        speak(tsay,008)         $$ "Symbols"
792   do        speak(tsay,009)         $$ "Words and Numbers Game"
793   do        speak(tsay,010)         $$ "Money"
```

```
794  do      speak(tsay,011)      $$ "Telephone"
795  do      speak(tsay,012)      $$ "Names"
796  do      speak(tsay,013)      $$ "Greetings"
797  do      speak(tsay,014)      $$ "This-That"
798  do      speak(tsay,015)      $$ "Past Tense"
799  do      speak(tsay,016)      $$ "Grammar"
800  do      speak(tsay,017)      $$ "Vocabulary"
801
802  do      speak(tsay,018)      $$ "When you choose which HELP you want..."
803  do      speak(tsay,019)      $$ "we will start that lesson over from the
804                               $$ beginning."
805
806  do      debug
807
808  ***** end of page 3 *****
809
810  jumpop page4
```

```
848
849  do      speak(tsay,024)          $$ "Simply means I will show you all the
850                                    $$ individual lessons and you can choose
851                                    $$ the one you want.  Watch!
852
853  do      lightpen(box5+33,uppery+1,ON)        $$ touch lightpen to lesson
854
855  do      flipbox(plesson)
856
857  do      screen(026)
858
859  do      debug
860
861  ***** SCREEN 27 *****
862
863  do      erasefld(balloon)
864
865  do      lessmenu(275,295)
866
867  at      balloonx+16,balloony-43
868  size    2
869  color   blue
870  write   It's just like
871          using a menu
872          in a restaurant.
873          You choose...
874
875  do      speak(tsay,025)          $$ "See?  It is just like using a menu in a
876                                    $$ restaurant.  This is what is available
877                                    $$ and you choose the one you want."
878
879  do      screen(027)
880
881  do      speak(tsay,026)          $$ "The lessons we will learn are:"
882  do      speak(tsay,027)          $$ "Alphabet"
883  do      speak(tsay,028)          $$ "Letters and..."
884  do      speak(tsay,029)          $$ "Words"
885  do      speak(tsay,030)          $$ "Numbers"
886  do      speak(tsay,031)          $$ "Words"
```

```
887  do      speak(tsay,032)       $$ "Symbols and..."
888  do      speak(tsay,033)       $$ "Words and Numbers Game"
889  do      speak(tsay,034)       $$ "Money"
890  do      speak(tsay,035)       $$ "Telephone"
891  do      speak(tsay,036)       $$ "Names"
892  do      speak(tsay,037)       $$ "Greetings"
893  do      speak(tsay,038)       $$ "This-That and..."
894  do      speak(tsay,039)       $$ "Past Tense"
895
896  do      speak(tsay,040)       $$ "So simple"
897  do      speak(tsay,041)       $$ "so easy"
898  do      speak(tsay,042)       $$ "such fun!"
899
900  do      debug
901
902  ***** SCREEN 28 *****
903
904  colore  black
905  erase                         $$ erase whole screen to get rid of lesson menu
906
907  do      picture(pcc,000,balloonx,balloony)      $$ put back talk balloon
908
909  do      boxes
910
911  do      speak(tsay,043)       $$ "Because I want you to feel comfortable
912                                 $$ with me."
913
914  at      balloonx+16,balloony-43
915  size    2
916  color   blue
917  write   I will now show
918          you 3 important
919          uses of your
920          keyboard.
921
922  do      speak(tsay,044)       $$ "I will now show you 3 important ways to
923                                 $$ use your keyboard to help you learn
924                                 $$ English."
925
```

```
926  do      screen(028)
927
928  do      debug
929
930  ***** SCREEN 29 *****
931
932  do      erasefld(balloon)
933
934  do      speak(tsay,045)          $$ "First..."
935
936  at      balloonx+24,balloony-43
937  size    2
938  color   blue
939  write   On the right of
940          your keyboard
941          is a key with
942          a STAR.
943
944  do      speak(tsay,046)          $$ "on the right of your keyboard is a key
945                                   $$ with a star..
946
947  do      picture(pcc,009,graphx+350,graphy)   $$ picture of a star.
948
949  do      speak(tsay,047)          $$ "It looks like this"
950
951  do      screen(029)
952
953  do      debug
954
955  ***** SCREEN 30 *****
956
957  do      erasefld(balloon)
958
959  do      speak(tsay,048)          $$ "everytime you want to tell me you have
960                                   $$ finished an exercise and want me to show
961                                   $$ you a new one."
962  do      speak(tsay,049)          $$ "hit the star key"
963
964  at      balloonx+32,balloony-57
```

```
965  size    2
966  color   blue
967  write   The English
968          word for star
969          is....
970
971  do      speak(tsay,50)          $$ "the English word for star is ..."
972
973  do      picture(pcc,010,graphx+16,graphy-35)      $$ star text
974
975  do      speak(tsay,051)         $$ "Star"
976  do      speak(tsay,052)         $$ "star is the word I will use from now on"
977  do      speak(tsay,053)         $$ "when you hit the star"
978  do      speak(tsay,054)         $$ "I will respond"
979
980  do      screen(030)
981
982  do      debug
983
984  ***** end of page 4 *****
985
986  do      speak(tclose)           $$ close entutor2.tf1
987
988  jumpop  page 5
```

```
 989   define  local
 990   index,1
 991   errorx = graphx
 992   errory = graphy-45
 993   width = 17
 994   sentence(24),1
 995   try,1                          $$ attempts counter for erase demo
 996   define    end
 997
 998   screen native
 999   spacing variable
1000   thick     on
1001
1002   if        debug=TRUE
1003   .         calc pictnam == 'spint'
1004   .         do   picture(pcc,000,balloonx,balloony)
1005   .         do   boxes
1006   .         do   speak(teninit)
1007   endif
1008
1009   zero      filespec
1010   pack      filespec,,\tesol\spint\entutor 3
1011   do        speak(topen)         $$ open entutor3.tfl
1012
1013   calc      try == 0             $$ reset demo attempts
1014
1015   *****SCREEN 31(36) *****
1016
1017   do        speak(tsay,000)      $$ "Second..."
1018
1019   1again                         $$ entry point for erase demo
1020
1021   do        erasefld(graph)
1022   do        erasefld(balloon)
1023
1024   at        balloonx+16,balloony-43
1025   size      2
```

```
1026   color     blue
1027   write     When you make
1028             a typing mistake
1029             it is easy to
1030             correct.
1031
1032   do        speak(tsay,001)      $$ "when you make a typing mistake"
1033   do        speak(tsay,002)      $$ "it's so easy to correct"
1034
1035   do        screen(031 + 5*try)
1036
1037   do        debug
1038
1039   ***** SCREEN 32 (37) *****
1040
1041   do        erasefld(balloon)
1042
1043   at        balloonx+48,balloony-43
1044   size      2
1045   color     blue
1046   write     A key with
1047             the English
1048             word ...
1049             ERASE.
1050
1051   do        screen(032 +  5*try)
1052
1053   do        speak(tsay,003)      $$ "On your Keyboard is a key with the
1054                                  $$ English word..."
1055
1056   do        picture(pcc,011,cluex,cluey-25)    $$ picture of the erase key.
1057
1058   do        speak(tsay,004)      $$ "ERASE"
1059
1060   do        speak(tsay,005)      $$ "Look at the word on your screen..."
1061
1062   do        speak(tsay,006)      $$ "Then find that key on your Keyboard."
1063
1064   do        speak(tsay,007)      $$ "Very good!"
```

```
1065
1066   do        speak(tsay,008)     $$ "In English"
1067
1068   do        speak(tsay,009)     $$ "ERASE is the word for erase"
1069
1070   do        speak(tsay,010)     $$ "ERASE is ERASE"
1071
1072   do        speak(tsay,011)     $$ "Each time you hit the ERASE key..."
1073
1074   do        speak(tsay,012)     $$ "you will erase one letter of what you
1075                                 $$ have typed"
1076
1077   do        speak(tsay,013)     $$ "you will ERASE starting at the right
1078                                 $$ end of what you have typed and moving
1079                                 $$ to the left."
1080
1081   do        speak(tsay,014)     $$ "It would be easier to understand if
1082                                 $$ you watched me correcting a typing mistake
1083   do        debug
1084
1085   ***** SCREEN 33 (38) *****
1086
1087   do        erasefld(balloon)
1088
1089   do        screen(033 + 5*try)
1090
1091   spacing   fixed
1092   thick     on
1093   size      2
1094   color     red+
1095   at        errorx,errory
1096   write     I luke learming English
1097
1098   pack      sentence(1),,I like learning English
1099
1100   do        speak(tsay,015)     $$ "I luke learming English."
1101
1102   do        speak(tsay,016)     $$ "I made two mistakes in spelling"
1103
```

```
1104   do        speak(tsay,017)      $$ "so I hit the erase key"
1105
1106   do        speak(tsay,018)      $$ "and erase"
1107
1108   do        speak(tsay,019)      $$ "one letter at a time"
1109
1110   loop      index == 22,3,-1
1111   .         at        errorx + width*index,errory
1112   .         erase     errorx+width*index,errory+28;errorx+width*(index+1),errory
1113   .         beep      0.1,400
1114   .         delay     0.1
1115   endloop
1116
1117   do        speak(tsay,020)      $$ "until I reach the first mistake"
1118
1119   do        speak(tsay,021)      $$ "I made"
1120
1121   do        speak(tsay,022)      $$ "I then retype the sentence correctly."
1122
1123   at        errorx+3*17,errory
1124   loop      index == 4,23
1125   .         showa     sentence(index)
1126   .         beep      0.1,400
1127   endloop
1128
1129   do        speak(tsay,023)      $$ "I like learning English."
1130
1131   do        debug
1132
1133   ***** SCREEN 34 (39) *****
1134
1135   do        erasefld(balloon)
1136
1137   do        screen(034 + 5*try)
1138
1139   do        speak(tsay,024)      $$ "That is all there is to it!
1140
1141   if        try=0
1142   .         do        debug
```

```
1143    .       branch    lnotagin                    $$ don't ask twice
1144    endif
1145
1146    do      speak(tsay,025)     $$ "Would you like to see that again?"
1147
1148    spacing variable
1149    size    2
1150    color   blue
1151    at      balloonx+48,balloony-43
1152    write   To see again
1153            touch your
1154            Lightpen to
1155            REPEAT.
1156
1157    do      speak(tsay,026)     $$ "Just touch you Lightpen to the REPEAT
1158                                $$ word."
1159
1160    do      lightpen(box3+33,uppery+1,ON)       $$ draw lightpen to repeat box
1161
1162    lwait
1163    do      pointer
1164    do      lightpen(box3+33,uppery+1,OFF)
1165    do      erasefld(graph)
1166    do      boxchek
1167    if      boxkey=prepeat
1168    .       branch lwait
1169    else
1170    .       calc      try == try + 1      $$ count this pass
1171    .       branch    lagain
1172    endif
1173
1174    ***** SCREEN 40 *****
1175
1176    lnotagin                              $$ entry to skip second demo of erase
1177
1178    do      erasefld(balloon)
1179    do      erasefld(clue)
1180    do      erasefld(graph)
1181
```

```
1182    do      screen(40)
1183
1184    do      speak(tsay,027)     $$ "it is so easy you won't have any
1185                                $$ trouble."
1186
1187    do      speak(tsay,028)     $$ "The third step is really exciting"
1188
1189    at      balloonx+32,balloony-43
1190    size    2
1191    color   blue
1192    write   How you can
1193            hear yourself
1194            speaking
1195            English.
1196
1197    do      speak(tsay,029)     $$ "It is how you can hear yourself
1198                                $$ speaking English."
1199    do      debug
1200
1201    ***** SCREEN 41 *****
1202
1203    do      erasefld(balloon)
1204
1205    do      screen(41)
1206
1207    do      speak(tsay,030)     $$ "On your Keyboard"
1208
1209    do      speak(tsay,031)     $$ "in the lower right-hand corner"
1210
1211    do      speak(tsay,032)     $$ "is a key with the English word"
1212
1213    at      balloonx+48,balloony-57
1214    size    2
1215    color   blue
1216    write   The talk Key
1217            looks like
1218            this...
1219
1220    do      picture(pcc,012,graphs+180,graphy)      $$ picture of talk key
```

```
1221
1222   do       speak(tsay,033)     $$ "TALK"
1223
1224   do       speak(tsay,034)     $$ "see the word on your screen"
1225
1226   do       speak(tsay,035)     $$ "Please find the TALK key on your
1227                                $$ Keyboard."
1228
1229   do       speak(tsay,036)     $$ "Very good!"
1230
1231   do       speak(tsay,037)     $$ "In English the TALK means talk."
1232
1233   do       speak(tsay,038)     $$ "Talk is TALK"
1234
1235   do       speak(tsay,039)     $$ "When I ask you to record a sentence"
1236
1237   do       speak(tsay,040)     $$ "hit the TALK key"
1238
1239   do       speak(tsay,041)     $$ "listen for a BEEP that sounds like this"
1240
1241   beep     0.5,200
1242
1243   do       speak(tsay,042)     $$ "then talk into your microphone."
1244
1245   do       speak(tsay,043)     $$ "Microphone is the English word for
1246                                $$ microphone."
1247
1248   do       speak(tsay,044)     $$ "After you have recorded your voice"
1249
1250   do       speak(tsay,045)     $$ "hit the TALK key again"
1251
1252   *do      speak(tsay,046)     $$ "then"
1253
1254   do       debug
1255
1256   ***** SCREEN 42 *****
1257
1258   do       erasefld(balloon)
1259   do       erasefld(graph)
```

```
1260
1261   do       screen(042)
1262
1263   do       speak(tsay,047)      $$ "on your Keyboard are two keys with
1264                                 $$ words in ENGLISH.
1265
1266   do       speak(tsay,048)      $$ "The words are LISTEN and SAVE.
1267
1268   do       speak(tsay,049)      $$ "See the words on your screen?"
1269
1270   at       balloonx+32,balloony-57
1271   size     2
1272   color    blue
1273   write    Listen and
1274            Save will
1275            look like this.
1276
1277   do       picture(pcc,013,graphx+16,graphy)       $$ listen
1278   do       picture(pcc,014,graphx+290,graphy)      $$ save key
1279
1280   do       speak(tsay,050)      $$ "Please find them on your Keyboard"
1281
1282   do       speak(tsay,051)      $$ "very good"
1283
1284   do       debug
1285
1286   ***** end of page 5 *****
1287
1288   do       speak(tclose)        $$ close entutor3.tfl
1289
1290   jumpop   page6
```

```
1291   screen    native
1292   spacing   variable
1293   thick     on
1294
1295   if        debug=TRUE
1296   .         calc      pictnam == 'spint'
1297   .         do        picture(pcc,000,balloonx,balloony)
1298   .         do        boxes
1299   .         do        speak(teninit)
1300   endif
1301
1302   zero      filespec
1303   pack      filespec,,\tesol\spint\entutor4          $$ open entutor4.tfl
1304   do        speak(topen)
1305
1306   ***** SCREEN 43 *****
1307
1308   do        erasefld(balloon)
1309
1310   at        balloonx+16,balloony-57
1311   size      2
1312   color     blue
1313   write     The English
1314             word for listen
1315             is LISTEN.
1316
1317   at        cluex+130,cluey-75
1318   size      2
1319   color     red+
1320   write     LISTEN
1321
1322   do        screen(43)
1323
1324   do        speak(tsay,000)      $$ "In English LISTEN is the word for
1325                                  $$ listen."
1326   do        speak(tsay,001)      $$ "Listen is LISTEN"
1327
```

```
1328    do      speak(tsay,002)     $$ "Hit the LISTEN key to hear yourself"
1329
1330    do      speak(tsay,003)     $$ "after you have listened"
1331
1332    do      debug
1333
1334    ***** SCREEN 44 *****
1335
1336    do      erasefld(clue)
1337    do      erasefld(balloon)
1338
1339    at      balloonx+32,balloony-57
1340    size    2
1341    color   blue
1342    write   The English
1343            word for save
1344            is SAVE.
1345
1346    at      cluex+130,cluey-75
1347    size    2
1348    color   red+
1349    write   SAVE
1350
1351    do      screen(44)
1352
1353    do      speak(tsay,004)     $$ "please hit the SAVE key"
1354
1355    do      speak(tsay,005)     $$ "to save what you have recorded"
1356
1357    do      speak(tsay,006)     $$ "In English SAVE is the word for save"
1358
1359    do      speak(tsay,007)     $$ "Save is SAVE."
1360
1361    do      debug
1362
1363    ***** SCREEN 45 *****
1364
1365    do      erasefld(clue)
1366    do      erasefld(balloon)
```

```
1367    do      erasefld(graph)
1368
1369    do      speak(tsay,008)     $$ "Once more quickly"
1370
1371    at      balloonx+32,balloony-57
1372    size    2
1373    color   blue
1374    write   You record
1375            with the TALK
1376            key and...
1377
1378    do      screen(45)
1379
1380    do      speak(tsay,009)     $$ "hit the TALK key"
1381
1382    do      speak(tsay,010)     $$ "listen for the BEEP"
1383
1384    beep    0.5,200
1385
1386    do      speak(tsay,011)     $$ "record your voice"
1387
1388    do      speak(tsay,012)     $$ "hit TALK again"
1389
1390    do      speak(tsay,013)     $$ "then hit the LISTEN key to hear your
1391                                $$ voice."
1392    do      speak(tsay,014)     $$ "Hit the SAVE key"
1393
1394    *do     speak(tsay,015)     $$ "then hit the STAR"
1395
1396    *do     speak(tsay,016)     $$ "so I can come back!"
1397
1398    do      speak(tsay,017)     $$ "It sounds more complicated than it
1399                                $$ really is."
1400
1401    do      speak(tsay,018)     $$ "You try it once and you'll discover how
1402                                $$ easily you can do it"
1403
1404    do      speak(tsay,019)     $$ "when I say NOW"
1405
```

```
1406    do          speak(tsay,020)     $$ "hit the TALK key"
1407
1408    do          speak(tsay,021)     $$ "after the BEEP"
1409
1410    do          speak(tsay,022)     $$ "record your voice"
1411
1412    do          speak(tsay,023)     $$ "Hit TALK again"
1413
1414    do          speak(tsay,024)     $$ "then hit LISTEN"
1415
1416    do          speak(tsay,025)     $$ "to hear yourself"
1417
1418    do          speak(tsay,026)     $$ "and last"
1419
1420    do          speak(tsay,014)     $$ "Hit the SAVE key"
1421    *do         speak(tsay,027)     $$ "hit SAVE and then STAR"
1422
1423    do          speak(tsay,028)     $$ "If you can't think of anything to say"
1424                                    $$ just say your name"
1425
1426    do          speak(tsay,029)     $$ "your address"
1427
1428    do          speak(tsay,030)     $$ "how you are feeling today"
1429
1430    do          speak(tsay,031)     $$ "anything at all"
1431
1432    do          speak(tsay,032)     $$ "Ready?"
1433
1434    do          speak(tsay,033)     $$ "Do it"
1435
1436    do          speak(tsay,034)     $$ "NOW!"
1437
1438    do          debug
1439
1440    ***** SCREEN 46 *****
1441
1442    do          erasefld(balloon)
1443    do          erasefld(graph)
1444
```

```
1445   nextop    page7
1446
1447   at        balloonx+32,balloony-57
1448   size      2
1449   color     blue
1450   write     Please record
1451             your voice
1452             NOW!!
1453
1454   do        screen(46)
1455
1456   at        graphx+100,graphy-28
1457   size      1
1458   color     white+
1459   write     When ready ...
1460             Hit TALK key to start recording sequence
1461             Wait for the beep
1462             Speak into the microphone loud and clear
1463             Hit TALK key again to stop recording
1464
1465   time      30                    $$ set timeout to 30 seconds
1466
1467   1talk
1468
1469   do        numlock               $$ make talk, listen, and save work
1470
1471   pause     keys=cins,timeup,next
1472
1473   if        zkey=timeup
1474   *.        do      speak(tsay,035)     $$ "Remember"
1475   *.        do      speak(tsay,036)     $$ "you must hit the STAR key"
1476   *.        do      speak(tsay,037)     $$ "or I can't come back."
1477   .         do      speak(tsay,038)     $$ "Please record your voice"
1478   .         do      speak(tsay,039)     $$ "or"
1479   .         do      speak(tsay,040)     $$ "if you have"
1480   .         do      speak(tsay,041)     $$ "hit the STAR key"
1481   .         do      speak(tsay,042)     $$ "NOW!"
1482   .         time    30
1483   .         branch  1talk
```

```
1484    elseif    zkey=cins
1485    .         branch    1talk
1486    endif
1487
1488    do        speak(trecord)
1489    pause                                        $$ kill the extra key
1490
1491    do        numlock
1492
1493    at        graphx+100,graphy-96
1494    size      1
1495    color     yellow
1496    write     Hit LISTEN key to hear your voice
1497
1498    pause     Keys=cdel,next
1499
1500    do        speak(tspeak)
1501
1502    at        graphx+100,graphy-110
1503    size      1
1504    color     green+
1505    write     Hit SAVE key to store your speech
1506
1507    pause     keys=cse,next
1508
1509    pack      filespec,,\tesol\save\save000.rec
1510    do        speak(tsave)                     $$ save the crap!
1511
1512    ***** end of page 7 *****
1513
1514    zero      filespec              $$ reset filespec due to save command
1515    pack      filespec,,\tesol\spint\entutor4
1516    do        speak(tclose)         $$ close entutor4.tfl
1517
1518    jumpop    page 7
```

```
1519   define    local
1520   helpx     = 325
1521   helpy     = 345
1522   define    end
1523
1524   screen    native
1525   spacing   variable
1526   thick     on
1527
1528   if        debug=TRUE
1529   .         calc      pictnam == 'spint'
1530   .         do        picture(pcc,000,balloonx,balloony)
1531   .         do        boxes
1532   .         do        speak(teninit)
1533   endif
1534
1535   do        erasefld(balloon)
1536   do        erasefld(graph)
1537
1538   zero      filespec
1539   pack      filespec,,\tesol\spint\entutor5
1540   dc        speak(topen)          $$ open entutor5.tfl
1541
1542   ***** SCREEN 47 *****
1543
1544   do        screen(047)
1545
1546   do        erasefld(balloon)
1547
1548   do        speak(tsay,000)       $$ "Did you have a good time?"
1549
1550   at        balloonx+64,balloony-43
1551   size      2
1552   color     blue
1553   write     I want you
1554             to enjoy
1555             learning
1556             English.
```

```
1557
1558    do          speak(tsay,001)     $$ "I hope so because I want you to enjoy
1559                                    $$ learning English."
1560    do          speak(tsay,002)     $$ "I have explained the mechanics of how
1561                                    $$ easily we will work together."
1562    do          speak(tsay,003)     $$ "now I want to explain how we will make
1563                                    $$ learning easier."
1564    do          speak(tsay,004)     $$ "I will teach you English words"
1565    do          speak(tsay,005)     $$ "one at a time"
1566    do          speak(tsay,006)     $$ "by saying the word to you in English
1567                                    $$ and Spanish"
1568    do          debug
1569
1570    ***** SCREEN 48 *****
1571
1572    do          screen(048)
1573
1574    do          erasefld(balloon)
1575
1576    at          balloonx+48,balloony-43
1577    size        2
1578    color       blue
1579    write       Writing the
1580                word in
1581                English and
1582                Spanish.
1583
1584    at          graphx+100,graphy-75
1585    size        2
1586    color       red+
1587    write       ENGLISH and SPANISH
1588
1589    do          speak(tsay,007)     $$ "and by writing the word on your screen
1590                                    $$ in English and Spanish"
1591    do          speak(tsay,008)     $$ "just like I am talking and writing now"
1592    do          speak(tsay,009)     $$ "this will be an easy way for you to
1593                                    $$ build your vocabulary of English words."
1594    do          speak(tsay,010)     $$ "We will review the English words we have
1595                                    $$ learned in each lesson."
```

```
1596    do          speak(tsay,011)     $$ "and you will have the opportunity to
1597                                    $$ answer questions to use your new English
1598                                    $$ words."
1599
1600    do          debug
1601
1602    ***** SCREEN 49 *****
1603
1604    do          screen(049)
1605
1606    do          erasefld(balloon)
1607    do          erasefld(graph)
1608
1609    at          balloonx+32,balloony-43
1610    size        2
1611    color       blue
1612    write       You can always
1613                review all the
1614                English words
1615                with HELP.
1616
1617    do          speak(tsay,012)     $$ "You can always review all the English
1618                                    $$ words you have learned by simply touching
1619                                    $$ the HELP lightbox and then using your
1620                                    $$ lightpen to touch VOCABULARY.
1621
1622    do          lightpen(box4+33,uppery+1,ON)
1623
1624    do          flipbox(phelp)
1625
1626    do          debug
1627
1628    ***** SCREEN 50 *****
1629
1630    do          erasefld(balloon)
1631
1632    at          balloonx+32,balloony-43
1633    size        2
1634    color       blue
```

```
1635    write       Then use your
1636                lightpen to
1637                select
1638                Vocabulary.
1639
1640    do          helpmenu(helpx,helpy)
1641
1642    do          screen(050)
1643
1644    do          debug
1645
1646    do          screen(999)
1647
1648    do          lightpen(helpx+10,helpy-264,ON)     $$ point at Vocabulary Review
1649
1650    do          vocab(265,345)      $$ pop up vocabulary review list
1651
1652    do          debug
1653
1654    ***** SCREEN 51 *****
1655
1656    colore      black
1657    erase
1658
1659    calc        pictnam == 'spint'          $$ for debug
1660    do          picture(pcc,000,balloonx,balloony)
1661    do          boxes
1662
1663    do          screen(051)
1664
1665    do          speak(tsay,013)     $$ "As your English vocabulary grows"
1666
1667    do          erasefld(balloon)
1668
1669    at          balloonx+16,balloony-57
1670    size        2
1671    color       blue
1672    write       We will begin
1673                using complete
```

```
1674            sentences.
1675
1676    do      speak(tsay,014)     $$ "we will begin using complete sentences"
1677
1678    at      graphx,graphy-75
1679    size    2
1680    color   red+
1681    write   You will learn English grammar.
1682
1683    do      speak(tsay,015)     $$ "and this is how you will learn English
1684                                $$ grammar."
1685    do      speak(tsay,016)     $$ "You will hear me speaking English words
1686                                $$ in the context of sentences"
1687    do      speak(tsay,017)     $$ "you will see the English words on your
1688                                $$ screen in the context of sentences."
1689    do      speak(tsay,018)     $$ "and you will have the opportunity to
1690                                $$ speak these sentences and type them
1691    do      speak(tsay,019)     $$ "and answer questions about them.
1692
1693    do      debug
1694
1695    ***** end of page 7 *****
1696
1697    jumpop    page8
```

```
1698   define    local
1699   helpx     = 325
1700   helpy     = 345
1701   define    end
1702
1703   screen    native
1704   spacing   variable
1705   thick     on
1706
1707   if        debug=TRUE
1708   .         calc      pictnam == 'spint'
1709   .         do        picture(pcc,000,balloonx,balloony)
1710   .         do        boxes
1711   .         do        speak(teninit)
1712   .         zero      filespec
1713   .         pack      filespec,,\tesol\spint\entutor5
1714   .         do        speak(topen)
1715   endif
1716
1717   ***** SCREEN 52 *****
1718
1719   do        screen(052)
1720
1721   do        erasefld(balloon)
1722   do        erasefld(graph)
1723
1724   at        balloonx+40,balloony-43
1725   size      2
1726   color     blue
1727   write     You can also
1728             use HELP to
1729             choose...
1730
1731   do        speak(tsay,020)     $$ "You can also always touch HELP with
1732                                 $$ you lightpen and then choose Grammar."
1733
1734   do        lightpen(box4+33,uppery+1,ON)
```

```
1735
1736    do          flipbox(phelp)
1737
1738    do          debug
1739
1740    ***** SCREEN 53 *****
1741
1742    do          screen(053)
1743
1744    at          balloonx+40,balloony-127
1745    size        2
1746    color       blue
1747    write       Grammar
1748
1749    do          helpmenu(helpx,helpy)
1750
1751    do          debug
1752
1753    do          screen(888)
1754
1755    do          lightpen(helpx+10,helpy-250,ON)    $$ point at Grammar Review
1756
1757    do          grammar(260,325)          $$ pop up grammar review list
1758
1759    do          debug
1760
1761    ***** SCREEN 54 *****
1762
1763    colore      black
1764    erase
1765
1766    do          screen(054)
1767
1768    calc        pictnam == 'spint'
1769    do          picture(pcc,000,balloonx,balloony)
1770    do          boxes
1771
1772    at          balloonx+40,balloony-43
1773    size        2
```

```
1774   color     blue
1775   write     Learning
1776             English will
1777             be fun and
1778             challenging.
1779
1780   do        speak(tsay,021)      $$ "Learning English will be fun and
1781                                  $$ challenging."
1782   do        speak(tsay,022)      $$ "Now"
1783   do        speak(tsay,023)      $$ "I believe you are ready to start the
1784                                  $$ lessons"
1785   do        speak(tsay,024)      $$ "so I am going to give you a choice"
1786   do        speak(tsay,025)      $$ "pick up your lightpen"
1787   do        debug
1788
1789   ***** SCREEN 55 *****
1790
1791   do        screen(055)
1792
1793   do        erasefld(balloon)
1794
1795   at        balloonx+16,balloony-43
1796   size      2
1797   color     blue
1798   write     Touch either
1799             "How to use
1800             your Keyboard"
1801             or CONTINUE.
1802
1803   do        helpmenu(helpx,helpy)    $$ pop up the help menu
1804
1805   do        speak(tsay,026)      $$ "and touch either the 'How to use your
1806                                  $$ Keyboard' box under HELP"
1807   do        speak(tsay,027)      $$ "or"
1808   do        speak(tsay,028)      $$ "if you already know 'How to use your
1809                                  $$ Keyboard"
1810   do        speak(tsay,029)      $$ "touch the CONTINUE word at the bottom
1811                                  $$ of your screen and we will begin our
1812                                  $$ first English lesson."
```

```
1813    do          speak(tsay,030)     $$ "Are you ready?"
1814    do          speak(tsay,031)     $$ "Alright"
1815    do          speak(tsay,032)     $$ "touch your lightpen to your choice"
1816    do          speak(tsay,033)     $$ "NOW"
1817
1818    do          pointer
1819
1820    do          boxchek
1821
1822    ***** Since the helpmenu pointer section is disabled, let's fake it.
1823    calc        helpkey == (helpy - lpy -32)/14 + 1
1824
1825    if          debug
1826    .           at          25:5
1827    .           size        1
1828    .           color       white
1829    .           write       The chump selected $$$
1830    .           if          boxkey=pcont
1831    .           .           write       CONTINUE
1832    .           elseif      helpkey=hlphowky
1833    .           .           write       How to use your Keyboard
1834    .           else
1835    .           .           write       something else
1836    .           endif
1837    .           do          debug
1838    endif
1839
1840    do          debug
1841
1842    if          boxkey=pcont
1843    .           jump        spair,start
1844    elseif      helpkey=hlphowky
1845    .           jump        enkey,start
1846    endif
1847
1848    jump        menu,start
1849
1850    ***** end of page 8 *****
```

```
1851   receive   field
1852
1853   colore    black
1854
1855   if        field  =  balloon
1856   .         colore    white+
1857   .         erase     balloonx+16, balloony-15;balloonx+256,balloony-127
1858   .         colore    black
1859   elseif    field  =  clue
1860   .         erase     cluex,cluey;639,answery+1
1861   elseif    field  =  quest
1862   .         erase     questx,questy;questx+256,questy-84
1863   elseif    field  =  answer
1864   .         erase     answerx,answery;answerx+352,answery-112
1865   elseif    field  =  input
1866   .         erase     inputx,inputy;inputx+256,inputy-28
1867   elseif    field  =  half
1868   .         colore    white+
1869   .         erase     balloonx+16,balloony-70;balloonx+256,balloony-127
1870   .         colore    black
1871   elseif    field  =  graph
1872   .         erase     graphx,graphy;graphx+graphxw,graphy-graphyw
1873   endif
```

```
1874   enable    pointer
1875
1876   pause     keys=pointer
1877
1878   if        debug=TRUE
1879   .         size      1
1880   .         colore    black
1881   .         color     white+
1882   .         thick     on
1883   .         if        zscreent=0
1884   .         .         at        20:20
1885   .         .         erase     20
1886   .         else
1887   .         .         at        25:60
1888   .         .         erase     20
1889   .         endif
1890   .
1891   .         if        zscreent=0
1892   .         .         at        20:20
1893   .         else
1894   .         .         at        25:60
1895   .         endif
1896   .         write     zpx = (s,zpx,3)
1897   .
1898   .         if        zscreent=0
1899   .         .         at        20:30
1900   .         else
1901   .         .         at        25:70
1902   .         endif
1903   .         write     zpy = (s,zpy,3)
1904   endif
1905
1906   calc      lpy == zpy + offsety
1907             lpx == zpx + offsetx
1908
1909   disable   pointer
```

```
1910   define    local
1911   xloc,2
1912   yloc,2
1913   offset    = 32
1914   cheight   = 14       $$ height of char in dots
1915   nchars    = 4        $$ number of chars in heading
1916   wchars    = 26       $$ number of chars in longest line
1917   nlines    = 18       $$ number of lines in the list
1918   width     = (wchars + 2)*9 $$ if thick is on, else ( )*8
1919   height    = nlines*cheight + offset + 6
1920   tonelen   = 0.1
1921   tonehz    = 800
1922   define    end
1923
1924   receive   xloc,yloc
1925   thick     on
1926   spacing   fixed
1927
1928   colore    cyan+
1929   erase     xloc, yloc; xloc+width,yloc-height
1930   color     blue+
1931   box       xloc,yloc;xloc+width,yloc-height;4
1932   size      2
1933   draw      xloc,yloc-offset;xloc+width,yloc-offset
1934   color     red+
1935   at        xloc+(width -  18*nchars)/2,yloc-offset + 2
1936   write     HELP
1937   size      1
1938   color     blue
1939   at        xloc+8,yloc-(offset+cheight)
1940   write     p Introduction
1941             p How to use your Keyboard
1942             ) Alphabet
1943               p Letters
1944               p Words
1945             ) Numbers
1946               p Words
```

```
1947                p Symbols
1948                p Words and Numbers Game
1949              p Money
1950              p Telephone
1951              p Names
1952              p Greetings
1953              p This-That
1954              p Past Tense
1955              p Grammar
1956              p Vocabulary
1957              p Don't want help   $$ must be the last line
1958
1959    spacing   variable
1960
1961    return                        $$ disable pointer for now
1962
1963    1main
1964
1965    do        pointer
1966
1967    calc      helpkey == (yloc - 1py -offset)/cheight + 1
1968
1969    if        debug=TRUE
1970    .         at          25:1
1971    .         erase       40
1972    .         at          25:1
1973    .         write       The man wants helpkey (s,helpkey)
1974    endif
1975
1976    if        helpkey     > nlines $or$ helpkey ≤ 0
1977    .         beep        tonelen,tonehz
1978    .         branch      1main
1979    elseif    helpkey     = nlines
1980    .         branch      1main
1981    *.        return
1982    elseif    helpkey     = h1pintro
1983    .         beep        tonelen,tonehz
1984    .         branch      1main
1985    .         do          xlocxx
```

```
1986    elseif    helpkey   = hlphowky
1987    .         beep      tonelen,tonehz
1988    .         branch    1main
1989    .         do        xlocxx
1990    elseif    helpkey   = hlpalpha
1991    .         beep      tonelen,tonehz
1992    .         branch    1main
1993    .         do        xlocxx
1994    elseif    helpkey   = hlpalflt
1995    .         beep      tonelen,tonehz
1996    .         branch    1main
1997    .         do        xlocxx
1998    elseif    helpkey   = hlpalfwd
1999    .         beep      tonelen,tonehz
2000    .         branch    1main
2001    .         do        xlocxx
2002    elseif    helpkey   = hlpnumbr
2003    .         beep      tonelen,tonehz
2004    .         branch    1main
2005    .         do        xlocxx
2006    elseif    helpkey   = hlpnumwd
2007    .         beep      tonelen,tonehz
2008    .         branch    1main
2009    .         do        xlocxx
2010    elseif    helpkey   = hlpsymbl
2011    .         beep      tonelen,tonehz
2012    .         branch    1main
2013    .         do        xlocxx
2014    elseif    helpkey   = hlpwngam
2015    .         beep      tonelen,tonehz
2016    .         branch    1main
2017    .         do        xlocxx
2018    elseif    helpkey   = hlpmoney
2019    .         beep      tonelen,tonehz
2020    .         branch    1main
2021    .         do        xlocxx
2022    elseif    helpkey   = hlptelep
2023    .         beep      tonelen,tonehz
2024    .         branch    1main
```

```
2025      elseif   helpkey   do      xlocxx
2026                         = hlpnames
```

```
2027        .       beep      toneien,tonehz
2028        .       branch    1main
2029        .       do        xlocxx
2030      elseif    helpkey = hlpgreet
2031        .       beep      tonelen,tonehz
2032        .       branch    1main
2033        .       do        xlocxx
2034      elseif    helpkey = hlpthth
2035        .       beep      tonelen,tonehz
2036        .       branch    1main
2037        .       do        xlocxx
2038      elseif    helpkey = hlppast
2039        .       beep      tonelen,tonehz
2040        .       branch    1main
2041        .       do        xlocxx
2042      elseif    helpkey = hlpgram
2043        .       beep      tonelen,tonehz
2044        .       branch    1main
2045        .       do        xlocxx
2046      elseif    helpkey = hlpvocab
2047        .       do        vocab(265,345)
2048        .       return
2049      endif
2050      define    local
2051      xloc,2
2052      yloc,2
2053      offset  = 32
2054      cheight = 14          $$ height of char in dots
2055      nchars  = 6           $$ number of chars in heading
2056      wchars  = 26          $$ number of chars in longest line
2057      nlines  = 13          $$ number of lines in the list
2058      width   = (wchars + 2)*9 $$ if thick is on, else (  ) *8
2059      height  = nlines*cheight + offset + 6
2060      toneleh = 0.1
2061      tonehz  = 800
2062      define  end
2063
2064      receive xloc,yloc
```

```
2065      thick    on
2066      spacing  fixed
2067
2068      color    cyan+
2969      erase    xloc,yloc;xloctwidth,yloc-height
2070      color    blue+
2071      box      xloc,yloc;xloctwidth,yloc-height;4
2072      size     2
2073      draw     xlocx,yloc-offset;xloclwidth,yloc-offset
2074      color    red+
2075      at       xloc+(width - 18*nchars)/2,yloc-offset
2076      write    LESSON
2077      size     1
2078      color
2079      at       xloc+8,yloc-(offset+cheight)
2080      write    ) Alphabet
2081               p Letters
2082               p Words
2083               ) Numbers
2084               p Words
2085               p Symbols
2086               p Words and Numbers Game
2087               p Money
2088               p Telephone
2089               p Names
2090               p Greetings
2091               p This-That
2092               p Past Tense
2093
2094      spacing variable
2095      define    local
2096      xloc,2
2097      yloc,2
2098      cheight = 14                $$ height of char in dots
2099      offset  = cheight + 4       $$ 2*cheight + 4 if size of heading is 2.
2100      nchars = 10                 $$ number of chars in heading
2101      wchars = 34                 $$ number of chars in longest line
2102      nlines = 15                 $$ number of lines in the list
2103      width  = (wchars + 2)*9     $$ if thick is on, else ( )*8
```

```
2104    height = nlines*cheight + offset + 6
2105    tonelen= 0.1
2106    tonehz = 800
2107    define   end
2108
2109    receive xloc,yloc
2110    thick    on
2111    spacing fixed
2112
2113    color cyan+
2114    erase    xloc, yloc;xloc+width,yloc-height
2115    color ->
2116    box      xloc,yloc;xloc+width,yloc-height;4
2117    draw     xloc,yloc-offset;xloc-width,yloc-offset
2118    color    red+
2119    size 1
2120    at       xloc+8,yloc-offset +2
2121    write    VOCABULARY      VOVCABULARIO
2122    color    blue
2123    at       xloc+8,yloc-(offset+cheight)
2124    write    English      Ing~
2125             headset      juego de aud!fonos
2126             keyboard     tablero de llaves
2127             lightpen     lightpen
2128             lightbox     lightbox
2129             continue     siga
2130             pause        haga pausa
2131             repeat       repita
2132             help         ayude
2133             lesson       lecc'n
2134             star         estrella
2135             erase        borrar
2136             talk         hoblar
2137             listen       escuchar
2138             save         guardar
2139
2140    spacing variable
2141    define    local
2142    xloc,2
```

```
2143    yloc,2
2144    cheight = 14                $$ height of char in dots
2145    offset  = 2*cheight + 4     $$ 2*cheight + 4 if size of heading is 2.
2146    nchars  = 14                $$ number of chars in heading
2147    wchars  = 35                $$ number of chars in longest line
2148    nlines  = 15                $$ number of lines in the list
2149    width   = (wchars + 2)*9    $$ if thick is on, else ( )*8
2150    height  = nlines*cheight + offset + 6
2151    tonelen = 0.1
2152    tonehz  = 800
2153    define    end
2154
2155    receive xloc,yloc
2156    thick     on
2157    spacing fixed
2158
2159    color   cyan+
2160    erase   xloc,yloc;xloc+width,yloc-height
2161    color   blue+
2162    box     xloc,yloc;xloc+width,yloc-height;4
2163    draw    xloc,yloc-offset;xloc+width,yloc-offset
2164    color   red+
2165    size    2
2166    at      xloc+(width -  1(*nchars)/2,yloc-offset + 2
2167    write   GRAMMAR REVIEW
2168    color   blue
2169    size    1
2170    at      xloc+8,yloc-(offset + cheight + 7*cheight)
2171    write   English grammar review appears here
2172
2173  spacing variable
2174    *  This is the English version of boxes
2175
2176    define  local
2177    boxnum, 1
2178    define  end
2179
2180    screen native
2181    size    1
```

```
2182
2183    nocheck review
2184    receive boxnum
2185
2186    *  This module displays the option boxes at the bottom of the screen
2187    *  CONTINUE,PAUSE,REPEAT,HELP,LESSON
2188
2189    if        boxnum=phone
2190    .         do        boxes(pcont)
2191    .         do        boxes(ppause)
2192    .         do        boxes(prepeat)
2193    .         do        boxes(phelp)
2194    .         do        boxes(plesson)
2195    elseif    boxnum=pcont          $$ Box 1 - CONTINUE - SIGA
2196    .         color     white+
2197    .         box       box1,uppery;box1+66,lowery;4
2198    .         color     blue+
2199    .         box       box112,upper-2;box1+64,lowery+2
2200    .         at        box1+17,texty
2201    .         color     cyan
2202    .         write     CONT
2203    elseif    boxnum=ppause         $$ Box 2 - PAUSE - PAUSA
2204    .         color     white+
2205    .         box       box2,uppery;box2+66,lowery;4
2206    .         color     green
2207    .         box       box2+2,uppery-2;box2+64,lowery+2
2208    .         at        box 2+13,texty
2209    .         color     blue
2210    .         write     PAUSE
2211    elseif    boxnum=prepeat $$ Box 3 - REPEAT - REPITA
2212    .         white+
2213    .         box       box3,uppery;box3+66,lowery;4
2214    .         color     red+
2215    .         box       box3+2,uppery-2;box3+64,lowery+2
2216    .         at        box3+9,texty
2217    .         color     blue
2218    .         write     blue
2219    elseif    boxnum=phelp          $$ Box 4 - HELP - AYUDE
2220    .         color     white+
```

```
2221    .       box     box4,uppery;box4+66,lowery;ds4
2222    .       color   brown+
2223    .       box     box4+2,uppery-2;box4+64,lowery+2
2224    .       at      box4+17,texty
2225    .       color   blue
2226    .       write   HELP
2227    elseif  boxnun=plesson      $$ box 5 - LESSON - LECCION
2228    .       color   white+
2229    .       box     box5,uppery;box5+66,lowery;4
2230    .       color   cyan
2231    .       box     box5+2,uppery-2;box5+64,lowery+2
2232    .       at      box5+5,texty
2233    .       color   blue
2234    .       write   LESSON
2235    endif
2236
2237    color   white+              $$ reset default foreground color
2238    * boxchek returns a value indicating which of the five boxes was selected
2239    * by the last pointer stroke
2240
2241    if      (zpy > uppery+offsety) $or$ (zpy < lowery+offsety)
2242    .       calc    boxKey == phone
2243    elseif  (zpx ≥ box1+offsetx $and$ zpx ≤ box1+66+offsetx)
2244    .       calc    boxKey == pcont
2245    elseif  (zpx ≥ box2+offsetx $and$ zpx ≤ box2+66+offsetx)
2246    .       calc    boxKey == ppause
2247    elseif  (zpx ≥ box3+offsetx $and$ zpx ≤ box3+66+offsetx)
2248    .       calc    boxKey == prepeat
2249    elseif  (zpx ≥ box4+offsetx $and$ zpx ≤ box4+66+offsetx)
2250    .       calc    boxKey == phelp
2251    elseif  (zpx ≥ box5+offsetx $and$ zpx ≤ box5+66+offsetx)
2252    .       calc    boxKey == plesson
2253    endif
2254    define  local
2255    boxnum,1
2256    define  end
2257
2258    screen  native
2259    size    1
```

```
2260    color     white+
2261    thick     on
2262
2263    nocheck   receive
2264    receive   boxnum
2265
2266    * This module displays the option boxes at the bottom of the screen
2267    * as does 'boxes' but this does them without color
2268    * CONTINUE,PAUSE,REPEAT,HELP,LESSON
2269    * SIGA, PAUSE,REPITA,AYUDE,LECCION
2270
2271    if        boxnum=phone
2272    .         do        flipbox(pcont)
2273    .         do        flipbox(ppause)
2274    .         do        flipbox(prepeat)
2275    .         do        flipbox(phelp)
2276    .         do        flipbox(plesson)
2277    elseif    boxnum=pcont          $$ Box 1 - CONTINUE - SIGA
2278    .         color     blue+
2279    .         box       box1,uppery;box1+66,lowery;4
2280    .         color     black
2281    .         box       box1+2,uppery-2;box1+64,lowery+2
2282    .         at        box1+17,texty
2283    .         color     white
2284    .         write     CONT
2285    elseif    boxnum=ppause         $$ Box 2 - PAUSE - PAUSA
2286    .         color     green
2287    .         box       box2,uppery;box2+66,lowery;4
2288    .         color     black
2289    .         box       box2+2,uppery-2;box2+64,lowery+2
2290    .         at        box2+13,texty
2291    .         color     white
2292    .         write     PAUSE
2293    elseif    boxnum=prepeat        $$ Box 3 - REPEAT - REPITA
2294    .         color     red+
2295    .         box       box3,uppery;box3+66,loweryt;4
2296    .         color     black
2297    .         box       box3+2,uppery-2;box3+64,lowery+2
2298    .         at        box3+9,texty
```

```
2299    .       color   white
2300    .       write   REPEAT
2301    elseif  boxnum=phelp        $$ Box 4 - HELP - AYUDE
2302    .       color   brown+
2303    .       box     box4,uppery;box4+66,lowery;4
2304    .       color   black
2305    .       box     box4+2,uppery-2;box4+64,lowery+2
2306    .       at      box4+17,texty
2307    .       color   white
2308    .       write   HELP
2309    elseif  boxnum=plesson      $$ Box 5 - LESSION - LECCION
2310    .       color   cyan
2311    .       box     box5,uppery;box5+66,lowery;4
2312    .       color   black
2313    .       box     box5+2,uppery-2;box5+64,lowery+2
2314    .       at      box5+5,texty
2315    .       color   white
2316    .       write   LESSON
2317    endif
2318
2319    color   white+              $$reset default foreground color
2320    define  local
2321    x,2
2322    y,2
2323    state,1
2324    define  end
2325
2326    thick   on
2327
2328    receive x,y,state
2329
2330    if      state=ON
2331    .       color   cyan
2332    .       draw    x,y;x,y+15;x-33,y+65;x-46,y+75;x-46,y+60;x-13,y+10;x,y
2333    .               ;;x-13,y+10;x,y+15;;x-46,y+60;x-33,y+65;;x-46,y+75;0,y+75
2334    elseif  state=ON
2335    .       color   black
2336    .       draw    x,y;x,y+15;x-33,y+65;x-46,y+75;x-46,y+60;x-13,y+10;x,y
2337    .               ;;x-13,y+10;x,y+15;;x-46,y+60;x-33,y+65;;x-46,y+75;0,y+75
```

```
endif
*  correct displays the happy face and words in balloon

spacing variable

do        erasefld(clue)

color     blue
size      2
at        balloonx+48,balloony-71
write     Your answer
          is correct

calc      pictnum == 'right'
do        picture(pcc,000,cluex,cluey-30)

*enable pointer
*1loop
*pause    Keys=pointer
*
*do       boxchek
*if       boxkey=phelp
*.        size 1
*.        at   25:40
*.        write     Select continue
*.        branch    1loop
*elseif   boxkey / pcont
*.        branch    1loop
*endif
*
*size     1
*at       25:40
*erase    20                    $$ get rid of help message

do        debug

do        erasefld(clue)
do        erasefld(balloon)
*  sad face displays the unhappy face and words in balloon
```

```
2377
2378    spacing variable
2379
2380    do          erasefld(clue)
2381
2382    color       blue
2383    size        2
2384    at          balloonx+48,balloony-71
2385    write       Your anser
2386                is wrong!
2387
2388    calc        pictnam == 'sad'
2389    do          picture(pcc,000,cluex,cluey-30)
2390
2391    *enable     points
2392    *lloop
2393    *pause      Keys=pointer
2394    *
2395    *do         boxchek
2396    *if         boxkey=phelp
2397    *.          size    1
2398    *.          at      25:40
2399    *.          write   Select continue
2400    *.          branch  lloop
2401    *elseif     boxkey / point
2402    *.          branch  lloop
2403    *endif
2404    *
2405    *size       1
2406    *at         25:40
2407    *erase      20                  $$ get rid of help message
2408
2409    do          debug
2410
2411    do          erasefld(clue)
2412    do          erasefld(balloon)
2413    do          erasefld(input)
2414    at          inputx,inputy-28
2415    size        1
```

```
2416    color     white+
2417    write     >
2418    define    local
2419    command,1
2420    entry,1
2421    leng = 0.1
2422    define    end
2423
2424    nocheck   receive
2425    receive command,entry
2426
2427    if        talk = FALSE  $$and$$ command =  trecord $and$ command =  speak $and$
                  command = tsave
2428    .         return
2429    endif
2430
2431    if        command = trecord
2432    .         delay    0.5        $$ avoid key bounce
2433    .         color     yellow
2434    .         at        25:71
2435    .         write     (blink,on)RECORDING(blink,off)
2436    .         color     white
2437    .         beep 0.5,2009
2438    endif
2439
2440    calc      al == command
2441              ah == entry
2442              bx == varloc(filespec)
2443    intcall   h70,tenspeak,tenspeak
2444    if        al = 0
2445    .         if        command - trecord
2446    .         .         beep   leng,800
2447    .         .         beep   leng,800
2448    .         .         beep   leng,800
2449    .         .         at     25:71
2450    .         .         erase  9
2451    .         endif
2452    .         return
2453    endif
```

```
2454
2455    if      al = 29                 $$ buffer overflow, don't print message
2456    .       beep    leng,500
2457    .       beep    leng,800
2458    .       beep    leng,500
2459    .       beep    leng,800
2460    .       beep    leng,500
2461    .       beep    leng,800
2462    .       return
2463    endif
2464
2465    size    1
2466    at      25:1
2467    write   TENSPEAK error (s,aL)    $$$
2468    write   aL;;;
2469            Invalid Function number;
2470            File not found;
2471            Path not found;
2472            Too many files open;
2473            Access denied
2474            Invalid handle;
2475            Memory control blocks destroyed;
2476            Insufficient memory;
2477            Invalid memory block address;
2478            Invalid environment;
2479            Invalid format;
2480            Invalid access code;
2481            Invalid data;
2482            Not defined;
2483            Invalid drive was specified;
2484            Attempt to remove current directory;
2485            Not same device;
2486            No more files;
2487            Illegal input file specification;
2488            Illegal index file;
2489            Illegal command number requested;
2490            Archive file is already open;
2491            Index too big for buffer;
2492            Entry number is out of range;
```

```
2493                    No speech in buffer;
2494                    Name did not match current archive;
2495                    No archive open;
2496                    VCLK limeout (hardware failure!);
2497                    Speech buffer overflow (warning);
2498                    Bad vox file format;
2499                    Illegal error returned by TENSPEAK - (show,a1)
2500    write   Command is "$$$
2501    writec  Command;;teninit;topen;tsay;tclose;tread;tspeak;tloud;trecord;tsave
2502    write   "$$$
2503    color   yellow
2504    at      25:76
2505    write   next
2506    pause   keys=next
2507    at      25:1
2508    color   black
2509    erase   80
2510    write   Entry
2511
2512    if      a1 = 0
2513    .       pause   keys=next
2514    .       exitsys
2515    endif
2516    define  local
2517    ii,2
2518    len = 20
2519    prog(len),1             $$ assembly language subroutine storage
2520    subec=,2
2521    highlow,2
2522    .       high,1
2523    .       low,1
2524    define  end
2525
2526    calc    highlow == varloc(savebx)
2527
2528    loop    ii == 1,13n
2529    .       calcs   ii-2,prog(ii) ==
2530    .               hle,            $$ push ds
2531    .       h0b8,00,00,             $$ mov ax,0000
```

```
2532    .       h,8e,h0d8,          $$ mov ds,ux
2533    .       h,0bb,h0c0,h01,     $$ mov bx,1C0H
2534    .       h8b,h07,            $$ mov ax,[bx]
2535    .       h1f,                $$ pop ds
2536    .       h0a3,low,high,      $$ mov [savebx],ax
2537    .       h0cb                $$retf
2538    endloop
2539
2540    asmcall prog(1),savebx
2541    calc    savebx == savebx $cls2$  8    $$exchange high, low bytes
2542
2543    if      savebx = h,00u0
2544    .       at          25:5
2545    .       write     TENSPEAK is not resident! Offset is (showh,savebx). $$$
2546    .                 Cannot proceed!!!  Press any key to exit
2547    .       pause     Keys=all
2548    .       exitsys   $$ Very important!  We must not call tenspeak if not loaded
2549    endif
2550    define  local
2551    picnumbr,2
2552    picname(32),1
2553    index,1
2554    piclen,2
2555    atx,2
2556    aty,2
2557    define  end
2558
2559    screen  native
2560
2561    nocheck receive
2562    receive type,picnumbr,atx,aty
2563
2564    packc   type;picname(1);piclen;;
2565            \tesol\spint\(a,pictnum)(t,picnumbr,3),pcx;
2566            \tesol\spint\(a,pictnum)(t,picnumbr,3),pcc
2567
2568    loop index == 1,32
2569    .       if      picname(index) = h20
2570    .       .       calc    picname(index) == h30
```

```
2571    .       endif
2572    .       if      picname(index) = h0d
2573    .       .       calc    picname(index) == 0
2574    .       endif
2575    endloop
2576
2577    *loop   index == 1,32
2578    *.      shown   picname(index)
2579    *outloop picname(index+1) = 0
2580    *endloop
2581
2582    if      type = pcc
2583    .       at      atx,aty
2584    endif
2585
2586    device  picture,plot,picname(1),piclen
2587    if      zreturn = -1
2588    .       size    1
2589    .       at      25:5
2590    .       write   Picture plotting error!!   $$$
2591    .       write   zreturn;;
2592    .               "picture.exe" not loaded;
2593    .               Invalid function code;
2594    .               DOS version <2;
2595    .               Can't open file;
2596    .               Not enough memory;
2597    .               Can't read file;
2598    .               No image stored;
2599    .               Wrong display type
2600    .       if      zreturn=3 $or$ zreturn=5
2601    .       .       write   "$$$
2602    .       .       loop    index == 1,32
2603    .       .       .       showa   picnume(index)
2604    .       .       outloop picname(index+1) = 0
2605    .       .       endloop
2606    .       .       write   "$$$
2507    .       endif
2608    .       pause   keys=next
2609    .       exitsys
```

```
2610    endif
2611    define    local
2612    screen,2
2613    define    end
2614
2615    if        debug = FALSE
2616    .         return
2617    endif
2618
2619    spacing fixed
2620
2621    receive screen
2622
2623    size      1
2624    color     white
2625    color     blue
2626    at        1:36
2627    erase     15
2628    at        1:36
2629    write     <<<SCREEN (s,screen)>>>
2630
2631    spacing variable
2632    color     black
2633    if        debug = FALSE
2634    .         return
2635    endif
2636
2537    size      1
2538    color     white
2539    at        1:73
2640    erase     7
2641    color     blue
2642    at        1:73
2643    write     *PAUSE*$$$
2644
2645    pause     keys=next,space
2646
2647    color     black
2648    at        1:73
```

```
2649    erase     7
2650    at        1:36
2651    erase     15
2652    define    local
2653    ii,2
2654    proglen = 17
2655    numlock(proglen),1
2656    define    end
2657
2658    loop      ii == 1,proglen
2659    .         calcs     ii-2,numlock(ii) ==
2660    .                   h1e,              $$ push ds
2661    .                   h0b8,00,00,       $$ mov ax,0000
2662    .                   h8e,h0d8,         $$ mov ds,ax
2663    .                   h0bb,h17,h04,     $$ mov bx,417H
2664    .                   h8a,h07,          $$ mov a1,[bx]
2665    .                   h24,h0df,         $$ and a1,0DFH
2666    .                   h88,h07,          $$ mov [bx],a1
2667    .                   h1f,              $$ pop ds
2668    .                   h0cb              $$ retf
2669    endloop
2670
2671    asmcall numlock(1)
```

-- entutor -- Unit Name Table -- Printed: 4/19/87 11.53 pm

boxchek	2238	boxes	2174	correct	2339	debug	2633
defines	1	erasfeld	1851	flipbox	2254	grammar	2141
helpmenu	1910	lessmenu	2050	lightpen	2320	numlock	2652
page2	393	page 3	640	page4	811	page5	989
page6	1291	page 7	1519	page8	1698	picture	2550
pointer	1874	sodface	2376	screen	2611	speak	2418
start	120	tenload	2516	vocab	2095		

APPENDIX B

1314395

```
                        PAGE ,132
                        TITLE TENSPEAK.ASM Hooks to TENCORE language for Voxcord
                        ;**********************
                        ;*** tenspeak.asm ***                    04-MAR-87
                        ;**********************
                        NAME tenspeak

                        ;adapted from tspeak.asm of 09-JAN-87
                        ;v2.1  modified on 07-APR-87 to fix speak_rec command
                        ;v2.2  modified on 09-APR-87 to add trecord, tsave,
                              tload commands
                        ;v2.3  fixed on 18-APR-87 jle bug in num_check: & handle
                              80 file tixs
                        ;v2.4  fix on 18-APR-87, name_check to allow path names

                        ;
                        ;Copyright INTECHNICA LEARNING SYSTEMS, Inc. 1987, 1988

                        ;TENSPEAK COMMANDS
                        ;teninit
                        ;a1=0                                  initialize TENSPEAK
                        ;
                        ;topen
                        ;a1=1, bx=OFFSET filespec.tfl          open,read,close.tix & open
                                                               .tfl
                        ;returns: cx = number_of_entries
                        ;
                        ;tsay
                        ;a1=2, ah=entry n                       read and speak entry n
                                                               of .tfl
                        ;
                        ;tclose
                        ;a1=3, bx=OFFSET filespec.tfl          close .tfl
```

```
;
;tread
;tspeak
;a1=5                                   speak the contents of
                                        previously loaded speech
;
;tload
;a1=6, bx=OFFSET filespec.rec           speak file filename.rec
;
;trecord
;a1=7                                   record speech into buffer
;                                       NOTE:  any key stops the
                                        recording
;tsave
;a1=8, bx=OFFSET filespec               save speech buffer in a
                                        file

;errors are returned in a1 (0 = no error)

;DOS errors
;error 1 - Invalid function number
;error 2 - File not found
;error 3 - Path not found
;error 4 - Too many files open
;error 5 - Access Denied
;error 6 - Invalid handle
;error 7 - Memory control blocks destroyed
;error 8 - Insufficient memory
;error 9 - Invalid memory block address
;error 10 - Invalid environment
;error 11 - Invalid format
;error 12 - Invalid access code
;error 13 - Invalid data
;error 14 - Not defined
;error 15 - Invalid drive was specified
;error 16 - Attempted to remove current directory
;error 17 - Not same device
;error 18 - No more files
;LOCAL errors
```

```
;error 19 - Illegal input file spec
;error 20 - Illegal index file
;error 21 - Illegal command number requested
;error 22 - archive file is already open
;error 23 - index too big for buffer
;error 24 - Entry number requested is out of range
;error 25 - No speech in buffer
;error 26 - Input filespec did not match current
           archive name
;error 27 - No archive open (tsay w/o topen)
;error 28 - VCLK timeout (Serious hardware failure
;error 29 - speech buffer overflow before switch depressed
;error 30 - bad rec file format

;Calling TENSPEAK under TENCORE
;
;tenspeak,8
;.     ah,1
;.     al,1
;.     bx,2
;filespec,45
;
;pack  filespec,,c:\tencore\tesol\talk\spint000
;calc  bx <= varloc(filespec)
;
;calc  al <= command 0 to 8
;intcall      h70,tenspeak,tenspeak
;jump  al;;;tenerror

;now allows input file specs of the following form:
;{drive}\directory1\directory2\directory 3\filename
 {.ext},[space,CR,0]

= 0028    MAX_SPEC        EQU     40      ;max length of
                                           filespec less
                                           ext + 1

= 0050    MAX_TIX         EQU     80          ;allow max
                                               of 80 tix
                                               entries
```

```
                        = 000C          SIZE_OF_TIX_ENTRY       EQU     12

                        = 03C0          TIX_BUFLEN              EQU     MAX TIX*SIZE_OF_TIX_ENTRY
                        = 000B          SIZE_OF_ENTRY_HEADER    EQU     11

                        = 0380          VOX_PORT        EQU     380h
                        = 0008          BYTE_SIZE       EQU     8
                        = 0004          SAMPLE_SIZE     EQU     4
                        = 0080          VMASK           EQU     80H
                        = 00C0          RESET_CMD       EQU     0C0H
                        = 0000          PLAY CMD        EQU     000H
                        = 0080          REC_CMD         EQU     080H
                        = FFF0          BUFFER_SIZE     EQU     0FFF0H
                        = 0060          IBMSTAT         EQU     60H
                        = 0060          IBMSTAT         EQU     60H

                                        ;MS-DOS FUNCTIONS CALLS

                        = 3C00          FCREATE         EQU     3C00H
                        = 3D00          FOPEN           EQU     3D00H
                        = 3E00          FCLOSE          EQU     3E00H
                        = 3F00          FREAD           EQU     3F00H
                        = 4000          FWRITE          EQU     4000H
                        = 4200          LSEEK           EQU     4200H

                        = 000D          CR              EQU     0DH
                        = 000A          LF              EQU     0AH

                        0000            code    SEGMENT byte public 'CODE'
                                                ASSUME cs:code,ds:data

                        0000            init_tenspeak PROC      NEAR

                        0000 8C C5                      mov     bp,es   ;save es
                        0002 B0 70                      mov al, 70h     ;get address of interrupt vector
                                                                         70h
                        0004 B4 35                      mov     ah,35h
                        0006 CD 21                      int     21h     ;es:bx = seg.off of int 70h
```

```
811  screen  native
812  spacing variable
813  thick   on
814
815  if      debug=TRUE
816  •       calc    pictnam == 'spint'
817  •       do      speak(teninit)
818  •       zero    filespec
819  •       pack    filespec,,\tesol\spint\entutor2
820  •       do      speak(topen)
821  endif
822
823  ***** SCREEN 26 *****
824
825  colore  black
826  erase                   $$ erase whole screen to get rid of help menu
827
828  do      picture(pcc,000,balloonx,balloony)        $$ put back talk balloon
829  do      boxes
830
831  do      speak(tsay,020)         $$ "At the right"
832  do      speak(tsay,021)         $$ "on the screen"
833  do      speak(tsay,022)         $$ "the lightbox word..."
834
835  do      flipbox(plesson)
836
837  do      speak(tsay,023)         $$ "LESSON"
838
839  at      balloonx+32,balloony-43
840  size    2
841  color   blue
842  write   LESSON simply
843          means I will
844          show you all
845          the lessons.
846
847  do      boxes(plesson)          $$ put back for demo
```

```
0008 0E                 push    cs
0009 1F                 pop     ds
000A 8B FB              mov     di,bx
000C BE 00A0 R          mov     si,OFFSET int_holder
000F FC                 cld
0010 B9 000A            mov     cx,10
0013 F3/ A6             rep     cmpsb
0015 75 0D              jnz     not_loaded

0017 BA 0066 R          mov     dx,OFFSET already_string
001A B4 09              mov     ah,09h
001C CD 21              int     21h
001E B0 01              mov     al,01
0020 B4 4C              mov     ah,4ch      ;exit with return code = 1
0022 CD 21              int     21h

0024                 not_loaded:
0024 B4 25              mov     ah,25h      ;setup new handler at int 70h
0026 B0 70              mov     al,70h
0028 BA 00A0 R          mov     dx,OFFSET int_handler
002B CD 21              int     21h

002D BA 0082 R          mov     dx,OFFSET signon_string
0030 B4 09              mov     ah,09h
0032 CD 21              int     21h

0034 B8 ---- R          mov     ax,data
0037 8E D8              mov     ds,ax
0039 C7 06 0000 R 0000  mov     filespec_ptr,0000
003F C7 06 00B2 R 0000  mov     length_of_speech,0000
0045 C7 06 0091 R 00FF  mov     tfl_handle,0FFH         ;in case teninit called
                                                        before topen

004B B1 04              mov     cl,4
004D BA 0476 R          mov     dx,OFFSET prog_end
0050 D3 EA              shr     dx,cl
0052 42                 inc     dx                  ;bump to next para
0053 81 C2 ---- R       add     dx,data
0057 89 16 00B4 R       mov     buffer_segment,dx       ;save address of 64K
```

```
                                                                  speech buffer
005B 2B D5                      sub     dx,bp           ;bp=es
005D 81 C2 1000                 add     dx,4096         ;64K speech buffer
0061 B8 3100                    mov     ax,3100h        ;terminate and remain resident
0064 CD 21                      int     21h

0066 0D 0A 54 45 4E 53 50  already_string DB CR,LF,'TENSPEAK ALREADY
                                         LOADED',CR,LF,''$'
     45 41 4B 20 41 4C 52
     45 41 44 59 20 4C 4F
     41 44 45 44 0D 0A 24
0082 0D 0A 54 45 4E 53 50  signon_string  DB CR,LF,'TENSPEAK V2.4
                                          NOW LOADED,CR,LF,'$'
     45 41 4B 2C 20 56 32
     2E 34 20 4E 4F 57 20
     4C 4F 41 44 45 44 0D
     0A 24

                           init_tenspeak   ENDP

00A0                       int_handler  PROC    FAR

00A0 FB                    sti
00A1 50                    push    ax              ;ax = [bp+16]
00A2 53                    push    bx              ;bx = [bp+14]
00A3 51                    push    cx              ;cx = [bp+12]
00A4 52                    push    dx              ;dx = [bp+10]
00A5 56                    push    si              ;si = [bp+08]
00A6 57                    push    di              ;di = [bp+06]
00A7 55                    push    bp              ;bp = [bp+04]
00A8 1E                    push    ds              ;ds = [bp+02]
00A9 06                    push    es              ;es = [bp+00]
00AA 8B EC                 mov     bp,sp
00AC BB ---- R             mov     ax,data
00AF 8E D8                 mov     ds,ax

00B1 8B 46 10              mov     ax,[bp+16]          ;get input command
00B4 3C 00                 cmp     al,00
00B6 75 03                 jnz     not_00
```

```
00B8 EB 4E 90                   jmp     teninit

00BB                    not_00:
00BB 3C 01                      cmp     al,01

00BD 75 03                      jnz     not_01
00BF EB 64 90                   jmp     topen

00C2                    not_01:
00C2 3C 02                      cmp     al,02
00C4 75 03                      jnz     not_02
00C6 E9 0253 R                  jmp     tsay

00C9                    not_02:
00C9 3C 03                      cmp     al,03
00CB 75 03                      jnz     not_03
00CD E9 0238 R                  jmp     tclose

00D0                    not_03:
00D0 3C 04                      cmp     al,04
00D2 75 03                      jnz     not_04
00D4 E9 01D1 R                  jmp     tread

00D7                    not_04:
00D7 3C 05                      cmp     al,05
00D9 75 03                      jnz     not_05
00DB E9 0269 R                  jmp     tspeak

00DE                    not_05
00DE 3C 06                      cmp     al,06
00E0 75 03                      jnz     not 06
00E2 E9 02F3 R                  jmp     tload

00E5                    not_06:
00E5 3C 07                      cmp     al,07
00E7 75 03                      jnz     not_07
                                jmp     trecord

00EC                    not_07:
```

03-

13144395

```
00EC 3C 08              cmp     al,08
00EE 75 03              jnz     not_08
00F0 E9 042B R          jmp     tsave

00F3                not_08:
00F3 B8 0015            mov     ax,21       ;'illegal function requested'
00F6 EB 03              jmp     SHORT error-exit

00F8                normal_exit:
00F8 B8 0000            mov     ax,0000     ;mp error exit point
00FB                error_exit:
00FB 89 46 10           mov     [bp+16]ax   ;put error return on the stock
00FE 07                 pop     es
00FF 1F                 pop     ds
0100 5D                 pop     bp
0101 5F                 pop     di
0102 5E                 pop     si
0103 5A                 pop     dx
0104 59                 pop     cx
0105 5B                 pop     bx
0106 58                 pop     ax
0107 CF                 iret
                    int_handler     ENDP

                    ;TENSPEAK initialization routine
                    ;       mov     al,0
                    ;       int     70

0108                teninit PROC    NEAR

0108 C7 06 0000 R 0000  mov     filespec_ptr,0000
010E C7 06 0002 R 000   mov     filespec_seg,0000
0114 8B 1E 0091 R       mov     bx,tfl_handle
0118 BB 3E00            mov     ax,FCLOSE
011B CD 21              int     21h
011D C7 06 0091 R 00FF  tfl_handle,0FFH         ;just in case
0123 EB D3              jmp     normal_exit

                    teninit ENDP
```

```
                        ;topen - opens filespec.tix, fills the tix_buffer
                         and opens the filespec.tfl
                        ;
                        ;       mov     al,1
                        ;       mov     bx,OFFSET asciz_filespec
                        ;       int 70

0125                    topen   PROC    NEAR

0125 83 3E 0000 R 00     cmp     filespec_ptr,0000
012A 74 05               jz      not_occupied
012C B8 0016             mov     ax,22           ;tix buffer already occupied
012F EB CA               jmp     error_exit

0131                    not_occupied:
0131 E8 0494 R           call    create_names    ;create tix & tfl names
0134 BA 0065 R           mov     dx,OFFSET name_tix
0137 B8 3D00             mov     ax,FOPEN
013A CD 21               int     21h
013C A3 0093 R           mov     tix_handle,ax
013F 73 02               jnc     open_tix_ok
0141 EB B8               jmp     error_exit

0143                    open_tix_ok
0143 8B 1E 0093 R        mov     bx,tix_handle
0147 B8 4200             mov     ax,LSEEK
014A B0 02               mov al,2                ;method 2
014C 33 C9               xor     cx,cx
014E 33 D2               xor     dx,dx
0150 CD 21               int     21h
0152 73 02               jnc     tix_seek_ok
0154 EB A5               jmp     error_exit

0156                    tix_seek_ok:
0156 83 F9 00                    cmp     cx,0000
0159 74 05                       jz      not_too_big
015B B8 0017             mov     ax,23           ;tix file too big for buffer
015E EB 9B               jmp     error_exit
```

```
0160                    not_too_big:
0160 81 FA 03C0                 cmp     dx,TIX_BUFLEN
0164 7E 05                      jle     not_so_big
0166 B8 0017                    mov     ax,23           ;tix file too big for buffer
0169 EB 90                      jmp     error_exit

016B                    not_so_big:
016B 8B 1E 0093 R               mov     bx,tix_handle   ;re-position at beginning of
                                                         file
016F B8 4200                    mov     ax,LSEEK
0172 33 C9                      xor     cx,cx
0174 33 D2                      xor     dx,dx
0176 CD 21                      int     21h
0178 73 03                      jnc     re_seek_ok
017A E9 00FB R                  jmp     error_exit

017D                    re_seek_ok:
017D 8D 16 00B6 R               lea     dx,tix_buffer
0181 B9 03C0                    mov     cx,TIX_BUFLEN
0184 B8 3F00                    mov     ax,FREAD
0187 CD 21                      int     21h
0189 73 03                      jnc     read_tix_ok
018B E9 00FB R                  jmp     error_exit

018E                    read_tix_ok:
018E A3 00A3 R                  mov     tix_length,ax
0191 8B 1E 0093 R               mov     bx,tix_handle
0195 B8 3E00                    mov     ax,FCLOSE
0198 CD 21                      int     21h
019A C7 0093 R 00FF             mov     tix_handle, OFFH  ;just in case

                        ;number of entries = tix_length/12

01A0 BA 0000                    mov     dx,0            ;clear high dividend
01A3 A1 00A3 R                  mov     ax,tix_length
01A6 B9 000C                    mov     cx,12
01A9 F7 F1                      div     cx
```

```
                        ;if dx (remainder) ! =0, then error

01AB 83 FA 00                   cmp     dx,0000
01AE 74 06                      jz      legal_tix
01B0 B8 0014                    mov     ax,20       ;"Illegal tix file"
01B3 E9 00FB R                  jmp     error_exit

01B6                    ;legal_tix:
01B6 A2 00A5 R                  mov     number_of_entries,al
01B9 32 E4                      xor     ah,ah
01BB 89 46 0C                   mov     [bp+12],ax  ;return number_of_entries in cx
                                                     register

01BE BA 0039 R                  mov     dx,OFFSET name_tfl
01C1 B8 3D00                    mov     ax,FOPEN
01C4 CD 21                      int     21h
01C6 A3 0091 R                  mov     tfl_handle,ax
01C9 73 03                      jnc     open_tfl_ok
01C9 E9 00FB R                  jmp     error_exit

01CE                    open_tfl_ok:
01CE E9 00F8 R                  jmp     normal_exit

                        topen   ENDP

                        ;tread reads entry n into buffer without saying it.  The
                         filespec is checked
                        ;to make sure it matches the currently open tfl file.

                        ;       mov     al,4
                        ;       mov     ah,entry_number
                        ;       mov     bx,OFFSET asciz_filespec
                        ;       int     70

01D1                    tread   PROC    NEAR

01D1 E8 055A R                  call    num_check   ;is entry_num in range?
01D4 E8 056F R                  call    name_check  ;does filespec match?
01D7 E8 01DD R                  call    read_only
01DA E9 00F8 R                  jmp     normal_exit
```

```
                        tread   ENDP

01DD                    read_only       PROC    NEAR

01DD B0 0C                      mov     al,SIZE_OF_TIX_ENTRY  ;fetch offset to
                                                               talk entry into
                                                               dx:cx

01DF F6 2E 00A6 R               imul    entry_num
01E3 8D 1E 00B6 R               lea     bx,tix_buffer
01E7 03 D8                      add     bx,ax
01E9 8B 4F 08                   mov     cx,[bx+08]  ;offset high
01EC 8B 57 0A                   mov     dx,[bx+10]  ;offset low
01EF 8B 1E 0091 R               mov     bx,tfl_handle
01F3 B8 4200                    mov     ax,LSEEK
01F6 CD 21                      int     21h
01F8 73 04                      jnc     seek_ok
01FA 5B                         pop     bx          ;get rid of return address off
                                                     stack
01FB E9 00FB R                  jmp     error_exit

01FE                    seek_ok:
01FE BA 00A7 R                  mov     dx,OFFSET entry_header
0201 B9 000B                    mov     cx,SIZE_OF_ENTRY_HEADER
0204 8B 1E 0091 R               mov     bx,tfl_handle
0208 B8 3F00                    mov     ax,FREAD
020B CD 21                      int     21h
020D 73 04                      jnc     read_tfl_ok
020F 5B                         pop     bx
0210 E9 00FB R                  jmp     error_exit

0213                    read_tfl_ok:
0213 8B 0E 00A8 R               mov     cx,length_of_record
0217 F8                         clc
0218 83 E9 0B                   sub     cx,SIZE_OF_ENTRY_HEADER
021B BA 0000                    mov     dx,0000         ;speech buffer offset
021E 8B 1E 0091 R               mov     bx,tfl_handle
0222 1E                         push    ds
0223 A1 00B4 R                  mov     ax,buffer_segment
```

```
0226 8E D8                      mov     ds,ax
0228 B8 3F00                    mov     ax,FREAD
022B CD 21                      int     21h
022D 1F                         pop     ds
022E 73 04                      jnc     read_rest_ok
0230 5B                         pop     bx
0231 E9 00FB R                  jmp     error_exit

0234                    read_rest_ok:
0234 A3 00B2 R                  mov     length_of_speech,ax
0237 C3                         ret

                        read_only       ENDP

                        ;tclose filespec - closes current .tfl file.  Filespec
                         is checked against
                        ;currently open .tfl file.

                        ;       mov     al,3
                        ;       mov     bx,OFFSET asciz_filespec
                        ;       int     70

0238                    tclose PROC     NEAR

0238 E8 056F R                  call    name_check
023B 8B 1E 0091 R               mov     bx,tfl_handle
023F B8 3E00                    mov     ax,FCLOSE
0242 CD 21                      int     21h
0244 C7 06 0000 R 0000          mov     filespec_ptr,0000
024A C7 06 0091 R 00FF          mov     tfl_handle,0FFH     ;make handle illegal
                                                             so any further
                                                            ;attempt to close handle
                                                             will fail.

0250 E9 00F8 R                  jmp     normal_exit

                        tclose ENDP

                        ;tsay n - read and say talk entry without filespec checking
                        ;       mov     al,2
```

```
                        ;       mov     ah,entry n
                        ;       int 70

                        ;This is short hand for:    mov     al,4
                                                    mov     ah,entry n
                                                    mov     bx,OFFSET asciz_filespec
                        ;                           int     70
                        ;                           mov     al,5

                        ;                           int     70

0253                    tsay    PROC    NEAR
0253 83 3E 000 R 00             cmp     filespec_ptr,0000
0258 75 06                      jnz     tsay_ok
025A B8 001B                    mov     ax,27               ;"No archive open
                                                              (tsay w/o topen)
025D E9 00FB R                  jmp     error_exit

0260                    tsay_ok:
0260 E8 055A R                  call    num_check   ;check for legal entry number
0263 E8 01DD R                  call    read_only   ;read talk entry into buffer
0266 EB 01 90                   jmp     tspeak      ;say it

                        tsay    ENDP

                        ;tspeak speaks a previously loaded speech entry
                        ;       mov     al,5
                        ;       int     70h

0269                    tspeak PROC     NEAR

0269 A1 00B2 R                  mov     ax,length_of_speech
026C 3D 0000                    cmp     ax,0000     ;if this is 0,then no message
                                                     resides in the
                                                    ;buffer and we better not try
                                                     to speak garbage.
026F 75 06                      jnz     buffer_occupied
0271 B8 0019                    mov     ax,25       ;"No message in buffer"
0274 E9 00FB R                  jmp     error_exit
```

```
0277                    buffer_occupied:

0277 BB 0000                    mov     bx,0000         ;offset to play buffer
027A BA 0380                    mov     dx,VOX_PORT     ;set dx for port address
027D BE 0004                    mov     si,SAMPLE_SIZE ; set si to bits per sample
0280 8E 06 00B4 R               mov     es,buffer_segment
0284 FA                         cli                     ;disable interrupts

                        ; activate play mode

0285 2B C9                      sub     cx,cx           ; set cx to count down to zero
0287                    pclk1o0:
0287 41                         inc     cx              ; cnt for time out if no v clk
0288 74 62                      jz      ep4rtn          ; when count rolls to 0 quit
                                                          rtn error
028A EC                         in      a1,dx           ; input status from vox card
028B A8 80                      test    a1,VMASK        ; test vclock bit
028D 2B C9                      jz      pclk1o0         ; loop while vclk low
028F 2B C9                      sun     cx,cx           ; set cx to count down to zero
0291                    pclk1o0:
0291 41                         inc     cx              ; cnt for time out if no v clk
0292 74 58                      jz      ep4rtn          ; when count rolls to 0 quit
                                                          rtn error
0294 EC                         in      a1,dx           ; input status from vox card
0295 A8 80                      test    a1,VMASK        ; test vclock bit
0297 75 F8                      jnz     pclkhi0         ; loop while vclk high
0299 B0 00                      mov     a1,PLAY_CMD     ; command to make vox card
                                                          play
029B EE                         out     dx,a1           ; output to vox card
029C                    pclko1
029C EC                         in      a1,dx           ; input status from vox card
029D A8 80                      test    a1,VAMSK        ; test vclock bit
029F 74 FB                      jz      pclk1o1         ; loop while vclk low

                        ; start of play

02A1 26: 8A 27                  mov     ah,BYTE PTR es:[bx] ; get first byte of data
02A4 43                         inc     bx              ; point ot next byte
```

```
02A5 BF 0008                    mov     di, BYTE_SIZE   ; set bits in byte
02A8 8B 0E 00B2 R               mov     cx,length_of_speech     ; set ct to length of
                                                                  buffer
02AC 49                         dec     cx              ; reduce buffer size
02AB EB 1B 90                   jmp     plyshift        ; start play at plyshift

                        plyloop:
02B0 D0 E8                      shr     a1,1            ; aline word for output
02B2 D0 E8                      shr     a2,2            ; shift right 1 bit shift in 0
02B4 D0 E8                      shr     a1,1
02B6 D0 EB                      shr     a1,1
02B6 D0 E8                      shr     a1,1            ; ok for 4 or 3 bit data
02B8 BE 0004                    mov     si,SAMPLE-SIZE  ; reset sample counter
02BB 50                         push    ax              ; save value in a1

02BC                    pclkhi2:
02BC EC                         in      a1,dx
02BD A8 80                      test    a1,VMASK        ; loop while vclk high
02BF 75 FB                      jnz     pclkhi2
02C1                    pclklo2:
02C1 EC                         in      a1,dx
02C2 A8 80                      test    a1,VMASK        ; loop while vclk low
02C4 74 FB                      jz      pclklo2

                        ; valid for output

02C6 58                         pop     ax              ; get valid data from stack
02C7 0C 00                      or      a1,PLAY_CMD
02C9 EE                         out     dx,a1           ; output to vox card
02CA                    plyshift:
02CA D0 DC                      rcr     ah,1            ; rotate data into carry bit
02CC D0 D8                      rcr     a1,1            ; rotate carry into a1
02CE 4F                         dec     di              ; reduce bits remaining per
                                                          byte
02CF 74 05                      jz      loaddata        ; if no more bits load byte
02D1 4E                         dec     si              ; reduce bits per sample
02D2 74 DC                      jz      plyloop         ; if end play sample
02D4 EB F4                      jmp     plyshift
```

```
02D6                    loaddata:                       ; load data byte & test buffer
                                                          length
02D6 26: 8A 27                mov     ah,BYTE PTR es:[bx]  ; get bye of packed data
02D9 43                       inc     bx                  ; point to next byte
02DA BF 0008                  mov     di,BYTE_SIZE        ; resets bits remaining per byte
02DD 49                       dec     cx                  ; reduce buffer size
02DE 74 05                    jz      endply              ; if more data continue
02E0 4E                       dec     si
02E1 74 CD                    jz      plyloop
02E3 EB E5                    jmp     plyshift
02E5                    endply:
02E5 FB                       sti                         ; enable interrupts
02E6 E8 03F6 R                call    init_vox
02E9 E9 00F8 R                jmp     normal_exit         ; all done

02EC                    ep4rtn:                         ; return with error.
02EC FB                       sti                         ; enable interrupts
02ED B8 001C                  mov     ax,28               ; VCLK timeout
02F0 E9 00FB R                jmp     error_exit          ; all done

                        tspeak ENDP

                        ;tload - reads and speaks individual .rec file
                        ;       mov     al,6
                        ;       mov     bx,OFFSET filename.rec
                        ;       int     70

02F3                    tload   PROC    NEAR

02F3 E8 04CF R                  call    create_rec_name ; process incoming.*.rec
                                                          filename
02F6 BA 000D R                  mov     dx,OFFSET name_rec
02F9 B8 3D00                    mov     ax,FOPEN
02FC CD 21                      int     21h
02FE 73 09                      jnc     open_rec_ok
0300 C7 06 0095 R 00FF          mov     rec_handle,0FFH
0306 E9 00FB R                  jmp     error_exit
```

```
0309                    open_rec_ok:
0309 A3 0095 R                  mov     rec_handle,ax
030C 8B D8                      mov     bx,ax
030E BA 00A7 R                  mov     dx,OFFSET entry_header
0311 B9 000B                    mov     cx,SIZE_OF_ENTRY_HEADER
0314 B8 3F00                    mov     ax,FREAD
0317 CD 21                      int     21h
0319 73 03                      jnc     read_rec_ok
031B E9 00FB R                  jmp     error_exit

031E                    read_rec_ok:
031E 3D 000B                    cmp     ax,SIZE_OF_ENTRY_HEADER
0321 72 07                      jb      bad_vox_file


0323 80 3E 00A7 R FF            cmp     BYTE PTR entry_header,0ffh    ; first byte
                                                                        better be
                                                                        0ffh

0328 74 06                      jz      vox_file_ok
032A                    bad_vox_file:
032A B8 001E                    mov     ax,30               ;"Bad vox file format"
032D E9 00FB R                  jmp     error_exit

0330                    vox_file_ok:
0330 8B 0E 00A8 R               mov     cx,length_of_record
0334 F8                         clc
0335 83 E9 0B                   sub     cx,SIZE_OF_ENTRY_HEADER
0338 BA 0000                    mov     dx,0000             ;speech buffer offset
033B 8B 1E 0095 R               mov     bx,rec_handle
033F 1E                         push    ds
0340 A1 00B4 R                  mov     ax,buffer_segment
0343 8E D8                      mov     ds,ax
0345 B8 3F00                    mov     ax,FREAD
0348 CD 21                      int     21h
034A 1F                         pop     ds
034B 73 03                      jnc     rec_rest_ok
034B E9 00FB R                  jmp     error_exit

0350                    rec_rest_ok:
```

```
0350 A3 00B2 R          mov     length_of_of_speech,ax
0353 B8 3E00            mov     ax,FCLOSE
035A CD 21              int     21h
035C 73 03              jnc     rec_close_ok
035E E9 00FB R          jmp     error_exit

0361                rec_close_ok:
0361 E9 0269 R          jmp     tspeak

                    tload  ENDP

                     ;trecords speech into the speech buffer
                     ;      mov      al,7
                     ;      int      70h
                     ;
                     ; the keyboard buffer is flushed of extra keys before
                       starting
                     ; any key on the keyboard will stop the recording

0364                 trecord PROC  NEAR
0364 E8 041E R          call    clear_inbuffer  ;clear any pending chars.

                    ; activate record mode

0367 BA 0380            mov     dx,VOX_PORT
036A 2B C9              sub     cx,cx           ; to zero cx
036C FA                 cli                     ; disable interrupts
036D                rclklo0:
036D 41                 inc     cx              ; cnt for time out if no v clk
036E 74 68              jz      errtn           ; when count rolls to 0 quit
                                                  rtn error
0370 EC                 in      al,VMASK        ; loop while vclk low
0373 74 F8              jz      rclklo0
0375 2B C9              sub     cx,cx           ; set cx to count down to zero
0377                rclkhi0
0377 41                 inc     cx              ; cnt for time out if no v clk
0378 74 5E              jz      errtn           ; when count rolls to 0 quit
                                                  rtn error
```

```
037A EC                 in      al,dx
037B A8 80              test    al,VMASK        ; loop while vclk high
037B 75 F8              jnz     rclkhi0
037F B0 80              mov     al,REC_CMD      ; must wait one clock
0381 EE                 out     dx,al           ; for valid data out
                    rclklo1:
0382 EC                 in      al,dx
0383 A8 80              test    al,VMASK        ; loop while vclk low
0385 74 FB              jz                      ; rclkliol

                    ; init for recording

0387 BF 0008            mov     di,BYTE_SIZE    ; set di to bits per byte
038A BB 0000            mov     bx,0000
038D B9 FFF0            mov     cx,BUFFER-SIZE  ; set cx to length of buffer
0390 8E 06 00B4 R       mov     es,buffer_segment
0394                recloop:                    ; start of record loop

                    ; check for end of record

0394 BA 0060            mov     dx,IBMSTAT
0397 EC                 in      al,dx
039A A8 60              test    al,IBMMASK
039A 75 2B              jnz     endrec
039C BA 0380            mov     dx,VOX_PORT
039F BE 0004            mov     si,SAMPLE_SIZE  ; set si to bits per sample
03A2                rclkhi2:
03A2 EC                 in      al,dx           ; read data at port
03A3 A8 80              test    al,VMASK
03A5                    jnz     rclkhi2         ; loop while V_CLK high
03A7                rclklo2:
03A7 EC                 in      al,dx
03A8 A8 80              test    al,VMASK        ; loop while V_CLK low
03AA 74 FB              jz      rclklo2

                    ; valid data

03AC                recschift:
03AC D0 D8              rcr     al, 1           ; rotate data into carry reg
```

```
03AE DO DC              rcr     ah, 1           ; rotate carry into ah
03B0 4F                 dec     di              ; reduce bits remaining per
                                                  byte
03B1 74 05              jz      savdata         ; if no more space save byte
03B3 4E                 dec     si              ; reduce bits per sample
03B4 74 DE              jz      recloop         ; if end of sample get new
                                                  sample
03B6 EB F4              jmp     recshift

03B8                savdata:                    ; save data byte & test buffer
                                                  length
03B8 26: 88 27          mov     BYTE PTR es:[bx],ah ; move packed data into
                                                      memory
03BB 43                 inc     bx              ; point to next byte
03BC BF 0008            mov     di,BYTE_SIZE    ; reset bits remaining in byte
03BF 49                 dec     cx              ; reduce buffer size
03C0 74 23              jz      over4rec        ; if more room continue
03C2 4E                 dec     si
03C3 74 CF              jz      recloop
03C5 EB E5              jmp     recshift
03C7                endrec:                     ; end of record
03C7 FB                 sti                     ; enable interrupts
03C8 E8 03F6 R          call    init_vox
03CB 89 1E 00B2 R       mov     length_of_speech,bx
03CF E8 03FD R          call    smooth
03D2 E8 041E R          call    clear_inbuffer  ;clear the key that stopped
                                                 the speech
03D5 E9 00F8 R          jmp     normal_exit     ; all done
03D8                errtn:                      ; return with error.
03D8 FB                 sti                     ; enable interrupts
03D9 C7 00B2 R 0000     mov     length of speech,0000
03DF B8 001C            mov     ax,28           ;VCLK timeout
03E2 E9 00FB R          jmp     error_exit      ; all done

03E5                 over4rec:
03E5 FB                 sti
03E6 E8 03F6 R          call    int_vox
03E9 89 1E 00B2 R       mov     length_of_speech,bx
```

```
03ED E8 03FD R          call    smooth
03F0 B8 001D            mov     ax,29           ;buffer overflow
03F3 E9 00FB R          jmp     error_exit

03F6                 init_vox:
03F6 BA 0380                mov     dx,VOX_PORT     ; address of vox card i/o port
03F9 B0 C0                  mov     al, RESET_CMD
03FB EE                     out     dx,al           ; set vox card to idle
03FC C3                     ret

03FD                 smooth:
03FD 8E 06 00B4 R           mov     es,buffer_segment
0401 BB 0000                mov     bx,0000
0404 B9 008                 mov     cx,8
0407 BF 0416 R              mov     di,OFFSET quiet4
040A                 smooth_it:
040A 2E: 8A 05              mov     al,cs:[di]
040D 26: 88 07              mov     BYTE PTR es:[bx],al
0410 43                     inc     bx
0411 47                     inc     di
0412 49                     dec     cx
0413 75 F5                  jnz     smooth_it
0415 C3                     ret
0416 80 80 80 80 80 80 80   quiet4 DB 80h,80h,80h,80h,80h,80h,80h,80h
     80

                    trecord ENDP

041E                clear_inbuffer:
041E B4 01                  mov     ah,1
0420 CD 16                  int     16h
0422 74 06                  jz      no_char_pending
0424 B4 00                  mov     ah,0
0426 CD 16                  int     16h
0428 EB F4                  jmp     clear_inbuffer

042A                no_char_pending:
042A C3                     ret
```

```
                        ;tsave - saves the speech buffer in a file
                        ;       mov     al,8
                        ;       int     70h
042B                    tsave   PROC    NEAR


042B 83 3E 00B2 R 00            cmp     WORD PTR length_of_speech,0000
0430 77 06                      ja      save_ok
0432 B8 0019                    mov     ax,25           ;"No message in buffer"
0435 E9 00FB R                  jmp     error_exit

0438                    save_ok:
0438 E8 04CF R                  call    create_rec_name
043B BA 000D R                  mov     dx,OFFSET name_rec
043E C6 06 00A7 R FF            mov     BYTE PTR entry_header, Offh
0443 B9 0000                    mov     cx,0000         ;file attribute
0446 B8 3C00                    mov     ax,FCREATE
0449 CD 21                      int     21h
0449 73 03                      jnc     create_ok
044D E9 00FB R                  jmp     error_exit

0450                    create_ok:
0450 A3 0095 R                  mov     rec_handle,ax

0453 A1 00B2 R                  mov     ax,length_of_speech
0456 05 000B                    add     ax,SIZE_OF_ENTRY_HEADER
0459 A3 00A8 R                  mov     length_of_record,ax

045C 8B 1E 0095 R               mov     bx,rec_handle
0460 8D 16 00A7 R               lea     dx,entry header
0464 B9 000B                    mov     cx,SIZE_OF_ENTRY_HEADER
0467 B8 4000                    mov     ax,FWRITE
046A CD 21                      int     21h
046C 8B 0E 00B2 R               mov     cx,length_of_speech
0470 BA 0000                    mov     dx,0000         ;ds:dx is offset to buffer
0473 8B 1E 0095 R               mov     bx,rec_handle
0477 1E                         push    ds
0478 A1 00B4 R                  mov     ax,buffer_segment
047B 8E D8                      mov     ds,ax
```

```
047D B8 4000                    mov     ax,FWRITE
0480 CD 21                      int     21h
0482 1F                         pop     ds
0483 73 03                      jnc     write_ok
0485 E9 00FB R                  jmp     error_exit

0488                    write_ok:
0488 B8 3E00                    mov     ax,FCLOSE
048B 8B 1E 0095 R               mov     bx,rec_handle
048F CD 21                      int     21h
0491 E9 00F8 R                  jmp     normal_exit

                        tsave   ENDP

0494                    create_names:
0494 8E 46 02                   mov     es,[bp+2]       ;get segment of filespec
0497 8C 06 0002 R               mov     filespec_seg,es ;save segment of input spec
049B 8B 76 0E                   mov     si,[bp+14]       get offset
049E 89 36 0000R                mov     filespec_ptr,si ;save offset to input file
                                                         spec

04A2 BF 0065 R          mov     di,OFFSET name_tix
04A5 E8 04F1 R          call    get_name        ;process incoming file name
04A8 BE 009B R          mov     si,OFFSET tix_ext
04AB 8C D8              mov     ax,ds
04AD 8E C0              mov     es,ax
04AF B9 0004            mov     cx,4
04B2 F3/ A4             rep     movsb
04B4 C6 05 00           mov     BYTE PTR [di],00 ;terminate string (make it
                                                  asciz)

04B7 33 C9              xor     cx,cx
04B9 8A CB              mov     cl,bl
04BB BE 0065 R          mov     si,OFFSET name_tix
04BE BF 0039 R          mov     di,OFFSET name_tfl
04C1 F3/ A4             rep     movsb
04C3 BE 0097 R          mov     si,OFFSET tfl_ext
04C6 B9 0004            mov     cx,4
04C9 F3/ A4             rep     movsb
04CB C6 05 00           mov     BYTE PTR[di],00
```

```
04CE C3                 ret

04CF                create_rec_name:
04CF 8B 76 0E               mov     si,[bp+14]
04D2 89 36 0008 R           mov     rec_ptr,si
04D6 8E 46 02               mov     es,[bp+02]
04D9 8C 06 000A R           mov     rec_seg,es
04DD BF 000D R              mov     di,OFFSET name_rec
04E0 E8 04F1 R              call    get_name
04E3 BE 009F R              mov     si,OFFSET rec_ext
04E6 1E                     push    ds
04E7 07                     pop     es
04E8 B9 0004                mov     cx,4
04EB F3/ A4                 rep     movsb
04ED C6 05 00               mov     BYTE PTR [di],00
04F0 C3                     ret

                    ;es = file name segment pointer
                    ;si = file name offset pointer
                    ;di = pointer to buffer for storage

04F1                get_name:
04F1 B1 09                  mov     cl,9
04F3 B3 00                  mov     bl,0                ;character counter
04F5 C6 06 000C R 00        mov     BYTE PTR drive_flag,00  ;only one drive
                                                            spec allowed

04FA                move_name:
04FA 26: 8A 04              mov     al,es:[si]
04FF 74 58                  jz      move_done
0501 3C 2E                  cmp     al,'.'
0503 74 54                  jz      move_done
0505 3C 0D                  cmp     al,CR
0507 74 50                  jz      move_done
0509 3C 00                  cmp     al,0
050B 74 4C                  jz      move_done
050D 3C 30                  cmp     al,'0'
050F 7C 2F                  jl      bad_name
0511 3C 39                  cmp     al,'9'
0513 7E 33                  jle     char_okay
```

```
0515 75 15                      jnz     not_drive
0519 80 3E 000C R 00            cmp     BYTE PTR drive_flag,00
051E 75 20                      jnz     bad_name
0520 C6 06 000C R FF            mov     BYTE PTR drive_flag,OFFH
0525 80 F9 08                   cmp     c1,8
0528 75 16                      jnz     bad_name
052A B1 0A                      mov     c1,10           ;reset name counter and allow
                                                         for ':'
052C EB 1A                      jmp     SHORT char_okay

052E                    not_drive:
052E 24 5F                      cmp     a1,'A'
0532 7C 0C                      jl      bad_name
0534 3C 5A                      cmp     a1,'Z'
0536 7E 10                      jle     char_okay
0538 3C 5C                      cmp     a1,'\'
053A 75 04                      jnz     bad_name
053C B1 0A                      mov     c1,10
053E EB 08                      jmp     SHORT char_okay

0540                    bad_name:
0540 58                         pop     ax              ;clear 'get_name' return
                                                         address
0541 58                         pop     ax              ;clear 'create_name' return
                                                         address
0542 B8 0013                    mov     ax,19           ;bad file name
0545 E9 00FB R                  jmp     error_exit

0548                    char_okay:
0548 88 05                      mov     BYTE PTR [di],al
054A 46                         inc     si
054B 47                         inc     di
054C FE C3                      inc     b1              ;count chars in name
054E 80 FB 28                   cmp     b1,MAX_SPEC     ;filespec overflow?
0551 7D ED                      jge     bad_name
0553 FE C9                      dec     c1
0555 75 A3                      jnz     move_name
0557 EB E7                      jmp     bad_name        ;max out chars before
                                                         terminator found
```

```
0559                    move_done:
0559 C3                         ret
055A                    num_check:
055A 88 26 00A6 R               mov     entry_num,ah
055E A0 00A6 R                  mov     al,entry_num
0561 3A 06 00A5 R               cmp     al,number_of_entries
0565 7C 07                      jl      entry_num_ok
0567 58                         pop     ax              ;get rid of return address off
                                                         stack
0568 B8 0018                    mov     ax,24           ;entry number requested is too
                                                         big
056B E9 00FB R                  jmp     error_exit

056E                    entry_num_ok
056E C3                         ret

                        ;compare input filespec name with current tix name.
                        ;Checks only name up to extension or terminator.

056F                    name_check:
056F 8E 46 02                   mov     es,[bp+02]      ;get input filespec segment
0572 8C 06 0006 R               mov     check_seg,es
0576 8B 76 0E                   mov     si,[bp+14]      ;get input filespec offset
0579 89 36 0004 R               mov     check_ptr,si
057D BF 0065 R                  mov     di,OFFSET name_tix
0580                    check_loop:
0580 8A 05                      mov     al,[di]
0582 3C 2E                      cmp     al,'.'          ;done with tix name?
0584 74 2C                      jz      check_done
0586 26: 8A 1C                  mov     bl,es:[si]
0589 80 FB 30                   cmp     bl,'0'
058C 7C 3D                      jl      mismatch
058E 80 FB 39                   cmp     bl,'9'
0591 7E 17                      jle     ok_to_compare
0593 80 FB 3A                   cmp     bl,':'
0596 74 12                      jz      ok_to_compare
0598 80 E3 5F                   and     bl,5fh
059B 80 FB 41                   cmp     bl,'A'
```

```
059E 7C 2B              jl      mismatch
05A0 80 FB 5A           cmp bl,'Z'
05A3 7E 05              jle     ok_to_compare
05A5 80 FB 5C           com     bl.'\'
05A8 75 21              jnz     mismatch
05AA                ok_to_compare:
05AA 3A C3              cmp     al,bl
05AC 75 1D              jnz     mismatch
05AE 46                 inc     si
05AF 47                 inc     di
05B0 EB CE              jmp     check_loop

05B2                check_done:             ;we hit '.' in name tix. Next char
                                             in filespec better
05B2 26: 80 3C 20       cmp     BYTE PTR es:[si],' ' ; be ' ' or '.' or CR or 0.
05B6 74 12              jz      match
05B8 26: 80 3C 2E       cmp     BYTE PTR es:[si],'.'
05BC 74 0C              jz      match
05BE 26: 80 3C 0D       cmp     BYTE PTR es:[si],CR
05C2 74 06              jz      match
05C4 26: 80 3C 00       cmp     BYTE PTR es:[si],0
05C8 75 01              jnz     mismatch
05CA C3             match: ret

05CB                mismatch:
05CB 58                 pop     ax              ;get rid of return address off
                                                 stack
05CC B8 001A            mov     ax,26           ;filespec did not match
05CF E9 00FB R          jmp     error_exit

;Look for file intro.vox on disk and play it if it exists.'

;play_intro:

;       mov     ax,SETDTA
;       mov     dx,OFFSET search_buf
;       int     21h
;       mov     ax,FILESEARCH
;       mov     cx,0                ;normal
```

```
                        ;       mov     dx,OFFSET intro_name
                        ;       int     21h
                        ;       jc      no_intro
                        ;       mov     dx,OFFSET intro_name
                        ;       call    load_vox
                        ;       call    play_msg
                        ;no intro:
                        ;ret

05D2                    code    ENDS

0000                    data    SEGMENT para public 'DATA'

0000 ????               filespec_ptr    DW      ?
0000 ????               filespec_seg    DW      ?
0004 ????               check_ptr       DW      ?
0006 ????               check_seg       DW      ?
0008 ????               rec_ptr         DW      ?
000A ????               rec_seg         DW      ?

000C ??                 drive_flag      DB      ?
000D 002C[              name_rec        DB      MAX_SPEC+4 DUP (0)
         00
             ]

0039 002C[              name_tfl        DB      MAX_SPEC+4 DUP (0)
         00
             ]

0065 002C[              name_tix        DB      MAX_SPEC+4 DUP (0)
         00
             ]

0091 ????               tfl_handle      DW      ?
0093 ????               tix_handle      DW      ?
0095 ????               rec_handle      DW      ?
0097 2E 54 46 4C        tfl_ext         DB      '.TFL'
009B 2E 54 49 58        tix_ext         DB      '.TIX'
009F 2E 52 45 43        rec_ext         DB      '.REC'
```

```
00A3 ????           tix_length      DW      ?
00A5 ??             number_of_entries       DB      ?
00A6 ??             entry_num               DB      ?

00A7 ??             entry_header            DB      ?       ;11 byte image of talk
                                                             file entry begins here
00A8 ????           length_of_record        DW      ?       ;length of absolute talk
                                                             entry
00AA 0008[                                  DB  8 DUP (?) ;name of talk entry
         ??
       ]
00B2 ????           length_of_speech        DW      ?       ;length of speech data
                                                             in talk entry
00B4 ????           buffer_segment          DW      ?

00B6 03C0[          tix_buffer              DB TIX_BUFLEN DUP (?)
         ??
       ]

= 0476                  prog_end        EQU     $

0476                    data            ENDS

                        END
```

Microsoft (R) Assembler Version 4.00

TENSPEAK.ASM Hooks to TENCORE language for VoxCard Symbols-1

Segments and Groups:

N a m e	Size	Align	Combine	Class
CODE	05D2	BYTE	PUBLIC	'CODE'
DATA	0476	PARA	PUBLIC	'DATA'

Symbols:

Name	Type	Value	Attr
ALREADY_STRING	L BYTE	0066	CODE
BAD_NAME	L NEAR	0540	CODE
BAD_VOX_FILE	L NEAR	032A	CODE
BUFFER_OCCUPIED	L NEAR	0277	CODE
BUFFER_SEGMENT	L WORD	00B4	DATA
BUFFER_SIZE	Number	FFF0	
BYTE_SIZE	Number	0008	
CHAR_OKAY	L NEAR	0548	CODE
CHECK_DONE	L NEAR	05B2	CODE
CHECK_LOOP	L NEAR	0580	CODE
CHECK_PTR	L WORD	0004	DATA
CHECK_SEG	L WORD	0006	DATA
CLEAR_INBUFFER	L NEAR	041E	CODE
CR	Number	000D	
CREATE NAMES	L NEAR	0494	CODE
CREATE_OK	L NEAR	0450	CODE
CREATE_REC_NAME	L NEAR	04CF	CODE
DRIVE_FLAG	L BYTE	000C	DATA

```
ENDPLY ..................  L NEAR    02E5        CODE
ENDREC ..................  L NEAR    03C7        CODE
ENTRY_HEADER ............  L BYTE    00A7        DATA
ENTRY_NUM................  L BYTE    00A6        DATA
ENTRY_NUM_OK ............  L NEAR    056E        CODE
EP4RTN ..................  L NEAR    02EC        CODE
ERROR_EXIT ..............  L NEAR    00FB        CODE
CRRTN ...................  L NEAR    03D8        CODE

FCLOSE ..................  Number    3E00
FCREATE .................  Number    3C00
FILESPEC_PTR.............  L WORD    0000        DATA
FILESPEC_SEG ............  L WORD    0002        DATA
FOPEN ...................  Number    3D00
FREAD ...................  Number    3F00
FWRITE ..................  Number    4000

GET_NAME ................  L NEAR    04F1        CODE

IBMMASK .................  Number    0060
IBMSTAT .................  Number    0060
```


TENSPEAK.ASM Hooks to TENCORE language for Voxcard Symbols-2

```
INIT_TENSPEAK ...........  N PROC    0000        CODE      Length = 00A0
INIT_VOX ................  L NEAR    03F6        CODE
INT_HANDLER .............  F PROC    00A0        CODE      Length = 0068

LEGAL_TIX ...............  L NEAR    01B6        CODE
LENGTH_OF_RECORD ........  L WORD    00A8        DATA
LENGTH_OF_SPEECH ........  L WORD    00A0
LF ......................  Number    000A
LOADDATA ................  L NEAR    02D6        CODE
LSEEK ...................  Number    4200

MATCH ...................  L NEAR    05CA        CODE
MAX_SPEC ................  Number    0028
MAX_TIX .................  Number    0050
```

MISMATCH	L NEAR	05CB	CODE	
MOVE_DONE	L NEAR	0559	CODE	
MOVE_NAME	L NEAR	04FA	CODE	
NAME_CHECK	L NEAR	056F	CODE	
NAME_REC	L BYTE	000D	DATA	Length = 002C
NAME_TFL	L BYTE	0039	DATA	Length = 002C
NAME_TIX	L BYTE	0065	DATA	Length = 002C
NORMAL_EXIT	L NEAR	00F8	CODE	
NOT_00	L NEAR	00BB	CODE	
NOT_01	L NEAR	00C2	CODE	
NOT_02	L NEAR	00C9	CODE	
NOT_03	L NEAR	00D0	CODE	
NOT_04	L NEAR	00D7	CODE	
NOT_05	L NEAR	00DE	CODE	
NOT_06	L NEAR	00E5	CODE	
NOT_07	L NEAR	00EC	CODE	
NOT_08	L NEAR	00F3	CODE	
NOT_DRIVE	L NEAR	052E	CODE	
NOT_LOADED	L NEAR	0024	CODE	
NOT_OCCUPIED	L NEAR	0131	CODE	
NOT_SO_BIG	L NEAR	016B	CODE	
NOT_TOO_BIG	L NEAR	0160	CODE	
NO_CHAR_PENDING	L NEAR	042A	CODE	
NUMBER_OF_ENTRIES	L BYTE	00A5	DATA	
NUM_CHECK	L NEAR	055A	CODE	
OK_TO_COMPARE	L NEAR	05AA	CODE	
OPEN_REC_OK	L NEAR	0309	CODE	
OPEN_TFL_OK	L NEAR	01CE	CODE	
OPEN_TIX_OK	L NEAR	0143	CODE	
OVER4REC	L NEAR	03E5	CODE	
PCLKHI0	L NEAR	0291	CODE	
PCLKHI2	L NEAR	02BC	CODE	
PCKLKO0	L NEAR	0287	CODE	
PCLKLO1	L NEAR	029C	CODE	
PCLKLO2	L NEAR	02C1	CODE	
PLAY_CMD	Number	0000		

```
PLYLOOP  ..............  L NEAR   02B0     CODE


TENSPEAK.ASM hooks to TENCORE language for VoxCard  Symbols-3

PLYSHIFT  .............  L NEAR   02CA     CODE
PROG_END  .............  NEAR     0476     DATA

QUIET4 ................  L BYTE   0416     CODE

RCLKH10 ...............  L NEAR   0377     CODE
RCLHK12 ...............  L NEAR   03A2     CODE
RCLKL00 ...............  L NEAR   036D     CODE
RICKL01 ...............  L NEAR   0382     CODE
RCLKL02 ...............  L NEAR   03A7     CODE
READ_ONLY .............  N PROC   01DD     CODE      Length = 005B
READ_REC_OK ...........  L NEAR   031E     CODE
READ_REST_OK ..........  L NEAR   0234     CODE
READ_TFL_OK ...........  L NEAR   0213     CODE
READ_TIX_OK ...........  L NEAR   018E     CODE
RECLOOP ...............  L NEAR   0394     CODE
RECSHIFT ..............  L NEAR   03AC     CODE
REC_CLOSE_OK ..........  L NEAR   0361     CODE
REC_CMD ...............  Number   0080
REC_EXT ...............  L BYTE   009F     DATA
REC_HANDLE ............  L WORD   0095     DATA
REC_PTR ...............  L WORD   0008     DATA
REC_REST_OK ...........  L NEAR   0350     CODE
REC_SEG ...............  L WORD   000A     DATA
RESET_CMD .............  Number   00C0
RE_SEEK_OK ............  L NEAR   017D     CODE

SAMPLE_SIZE ...........  Number   0004
SAVDATA ...............  L NEAR   03BB     CODE
SAVE_OK ...............  L NEAR   0438     CODE
SEEK_OK ...............  L NEAR   01FE     CODE
SIGNON_STRING .........  L BYTE   0082     CODE
SIZE_OF_ENTRY_HEADER ..  Number   000B
SIZE_OF_ENTRY .........  Number   000C
```

```
SMOOTH ..................  L NEAR    03FD      CODE
SMOOTH_IT ...............  L NEAR    040A      CODE

TCLOSE ..................  N PROC    0238      CODE      Length = 001B
TENINIT .................  N PROC    0108      CODE      Length = 001D
TFL_EXT .................  L BYTE    0097      DATA
TFL_HANDLE ..............  L WORD    0091      DATA
TIX_BUFFER ..............  L BYTE    00B6      DATA      Length = 03C0
TIX_BUFLEN ..............  Number    03C0
TIX_EXT .................  L BYTE    009B      DATA
TIX_HANDLE ..............  L WORD    0093      DATA
TIX_LENGTH ..............  L WORD    00A3      DATA
TIX_SEEK_OK .............  L NEAR    0156      CODE
TLOAD ...................  N PROC    02F3      CODE      Length = 0071
TOPEN ...................  N PROC    0125      CODE      Length = 00AC
TREAD ...................  N PROC    01D1      CODE      Length = 000C
TRECORD .................  N PROC    0364      CODE      Length = 00BA
TSAVE ...................  N PROC    0428      CODE      Length = 0069
TSAY ....................  N PROC    0253      CODE      Length = 0016
TSAY_OK .................  L NEAR    0260      CODE
```



```
TSPEAK ..................  N PROC    0269      CODE      Length = 008A

VMASK ...................  Number    0080
VOX_FILE_OK .............  L NEAR    0030      CODE
VOX-PORT ................  Number    0380

WRITE_OK ................  L NEAR    0488      CODE
```

1071 Source Lines
1071 Total Lines
 163 Symbols

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. An interactive instruction apparatus comprising:

video display means for presenting text and graphic messages selected to exercise student reading and comprehension skills;

audio output means for presenting audio messages selected to exercise student listening skills;

audio input means for receiving audio responses selected to exercise student speaking skills;

text input means for receiving text responses selected to exercise student writing skills;

student speech reproduction means for receiving, digitizing and reproducing a student speech response;

reference response generation means for generating a reference speech response from a digital recording;

exercise generating means for generating a series of exercise, said exercises comprising: the presentation of text and graphic messages through the video display means, the presentation of audio messages through the audio output means, the reception of text responses through the text input means and the reception of audio responses through the audio input means.

at least one of said exercises including a text message for prompting a student speech response, an interactive period during which the student speech reproduces receives and reproduces a student speech response and the reference response

generation means generates a reference response in comparative relation with the student response; and

exercise control means responsive to the student for either 1) autonomously signalling the exercise generating means to generate an exercise, or alternatively 2) signalling the exercise generating means to repeat an interactive period.

2. An apparatus as in claim 1 wherein:

the message presenting means includes a plurality of presenting means, each for presenting a message to a student asynchronously with other presenting means;

the student speech reproduction means includes means for receiving and reproducing a plurality of asynchronous student speech responses;

the reference response generation means includes means for generating a plurality of reference responses, each in comparative associative relation with a reproduced student speech response;

the exercise control means includes means responsive to each of a plurality of students for either 1) autonomously signalling the exercise generating means to generate an exercise for a student asynchronously with exercises for other students, or alternatively 2) signalling the exercise generating means to repeat an interactive period for an exercise for a student asynchronously with interactive period for other students.

3. An apparatus as in claim 1 wherein audio and video messages of an exercise simultaneously symbolize a student speech response associated with that exercise.

4. An apparatus as in claim 1 further comprising student response storage means for recording a student speech response of a medium separable from the apparatus.

5. An interactive instruction apparatus comprising:

message presenting means for presenting text and graphic visual messages and for presenting audio messages of an exercise to a student;

message receiving means for receiving text and audio responses from a student;

student speech reproduction means for receiving and reproducing a student speech response;

reference response generation means for generating a reference speech response; and

exercise generating means connected to the message presenting means, to the message receiving means, to the student speech reproduction means and to the reference response generation means for generating a series of exercises, said exercises comprising the presentation of text, graphic, and audio messages, and the reception of text and audio responses;

wherein at least one of said exercises includes a message for prompting a student speech response, and an interactive period for receiving and reproducing a student speech response in comparative relation with a reference speech response.

Ridout & Maybee
101 Richmond St. West
Toronto, Canada M5H 2J7
Patent Agents of the Applicant

Fig. 1.

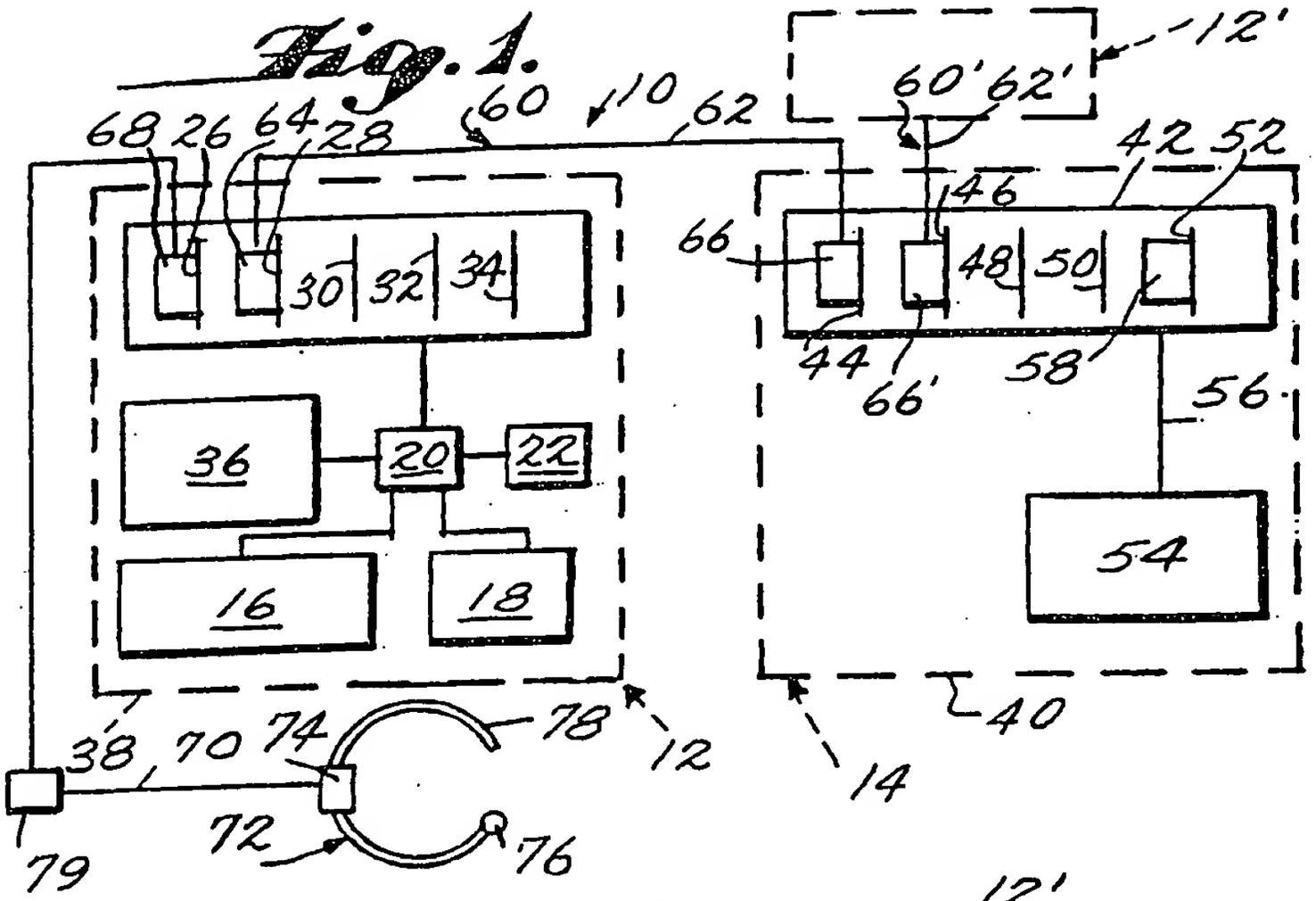
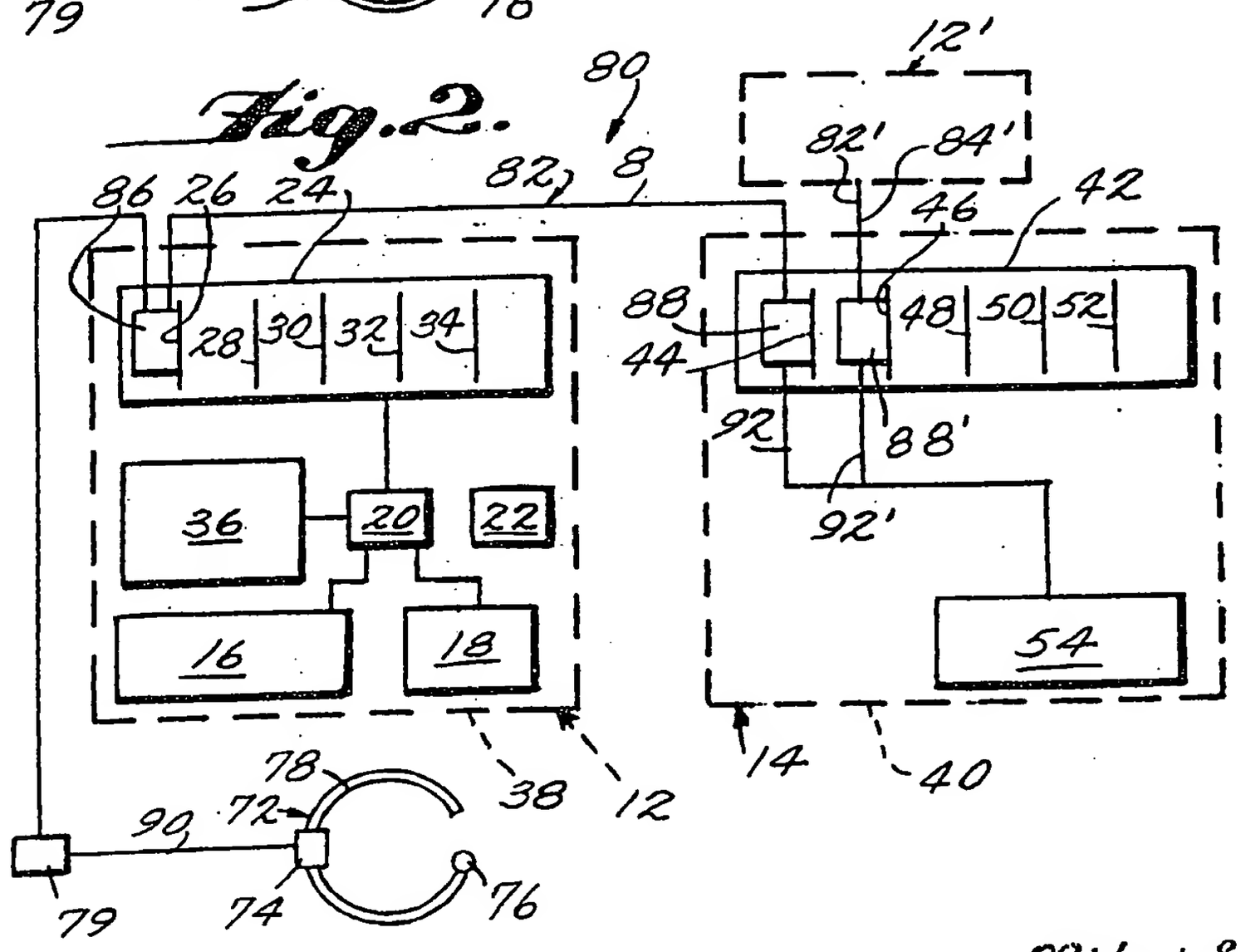


Fig. 2.



Ridout & Maybee
PATENT AGENTS

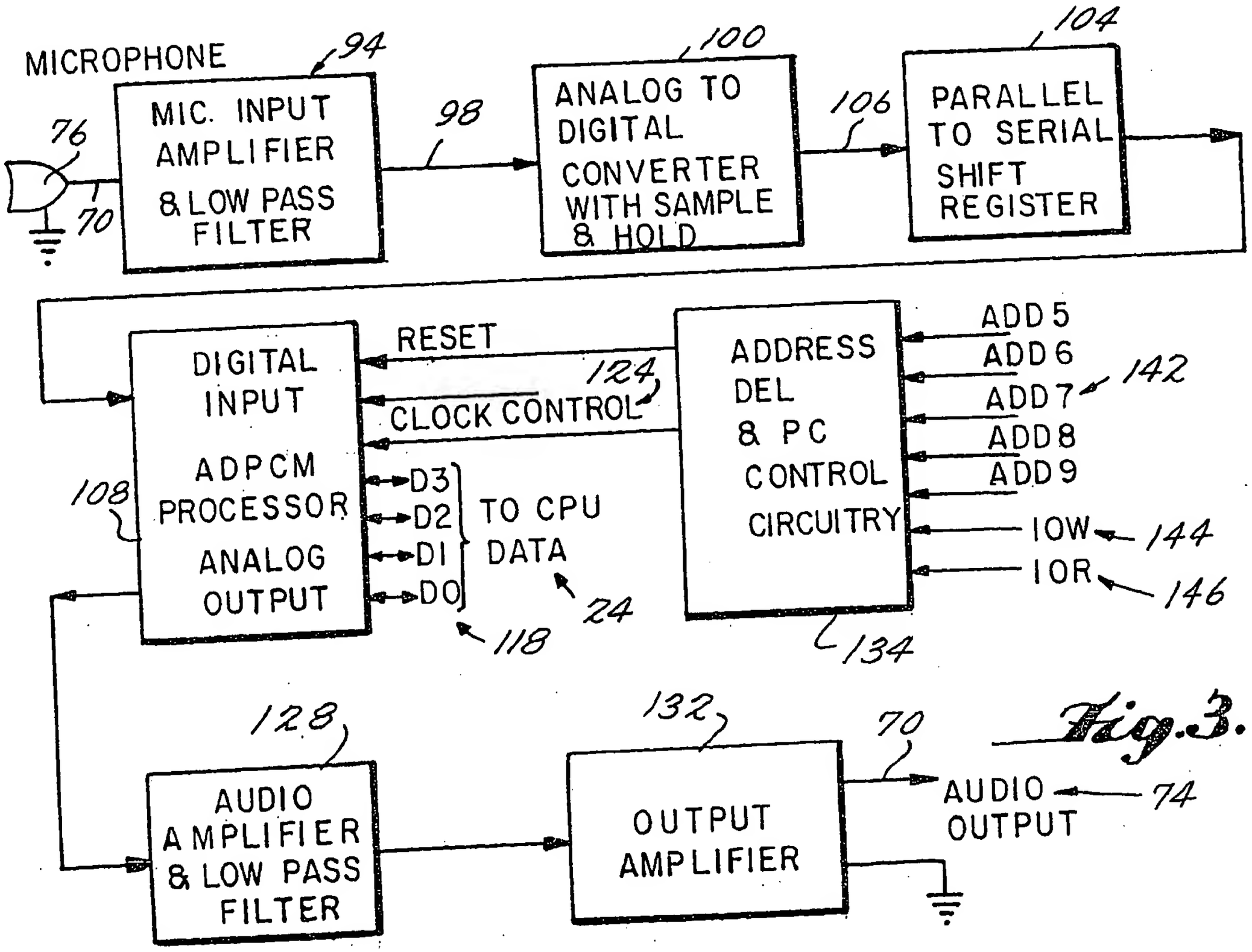
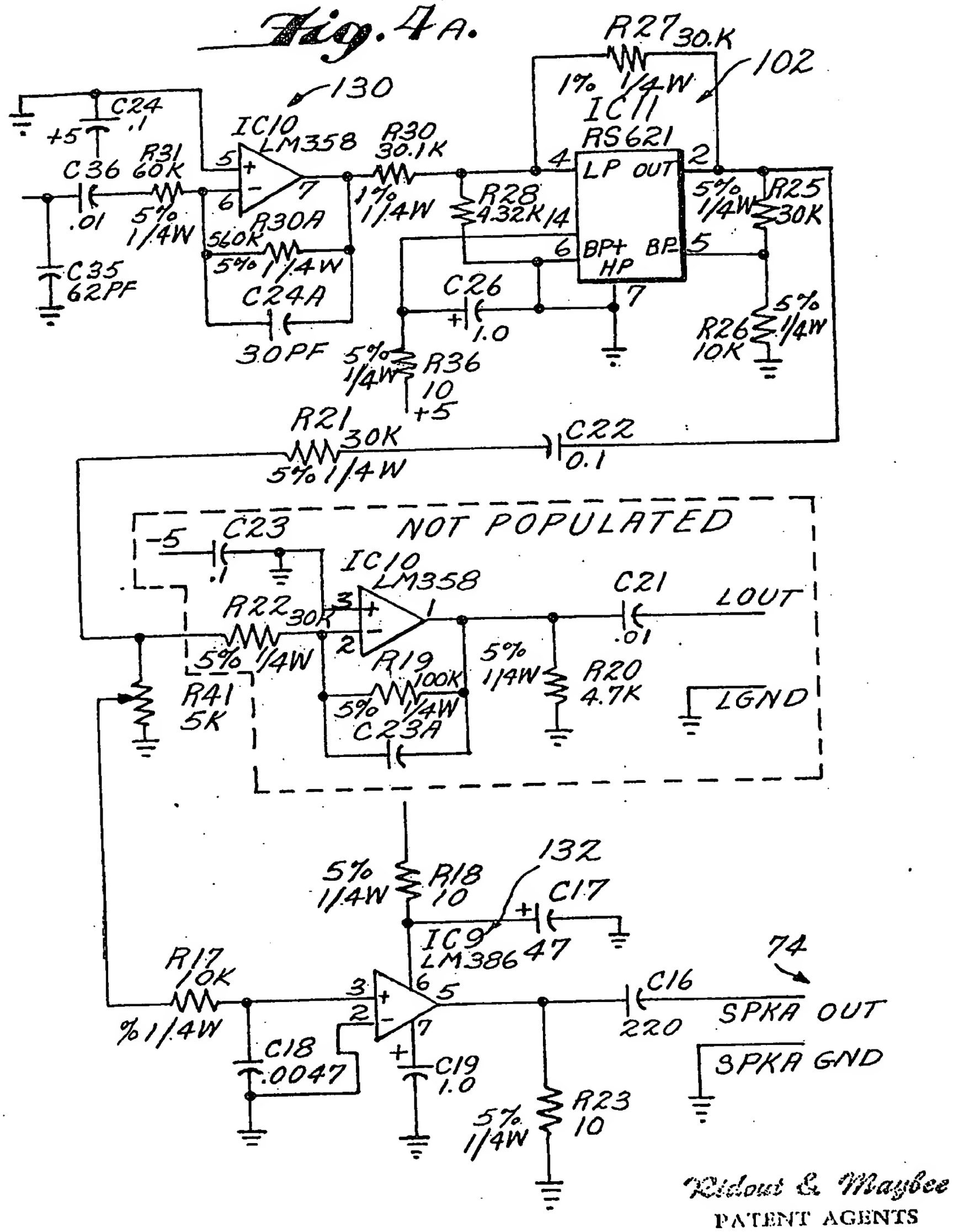


Fig. 3.

Robert E. Hayden
PATENT AGENTS

1314395
1912

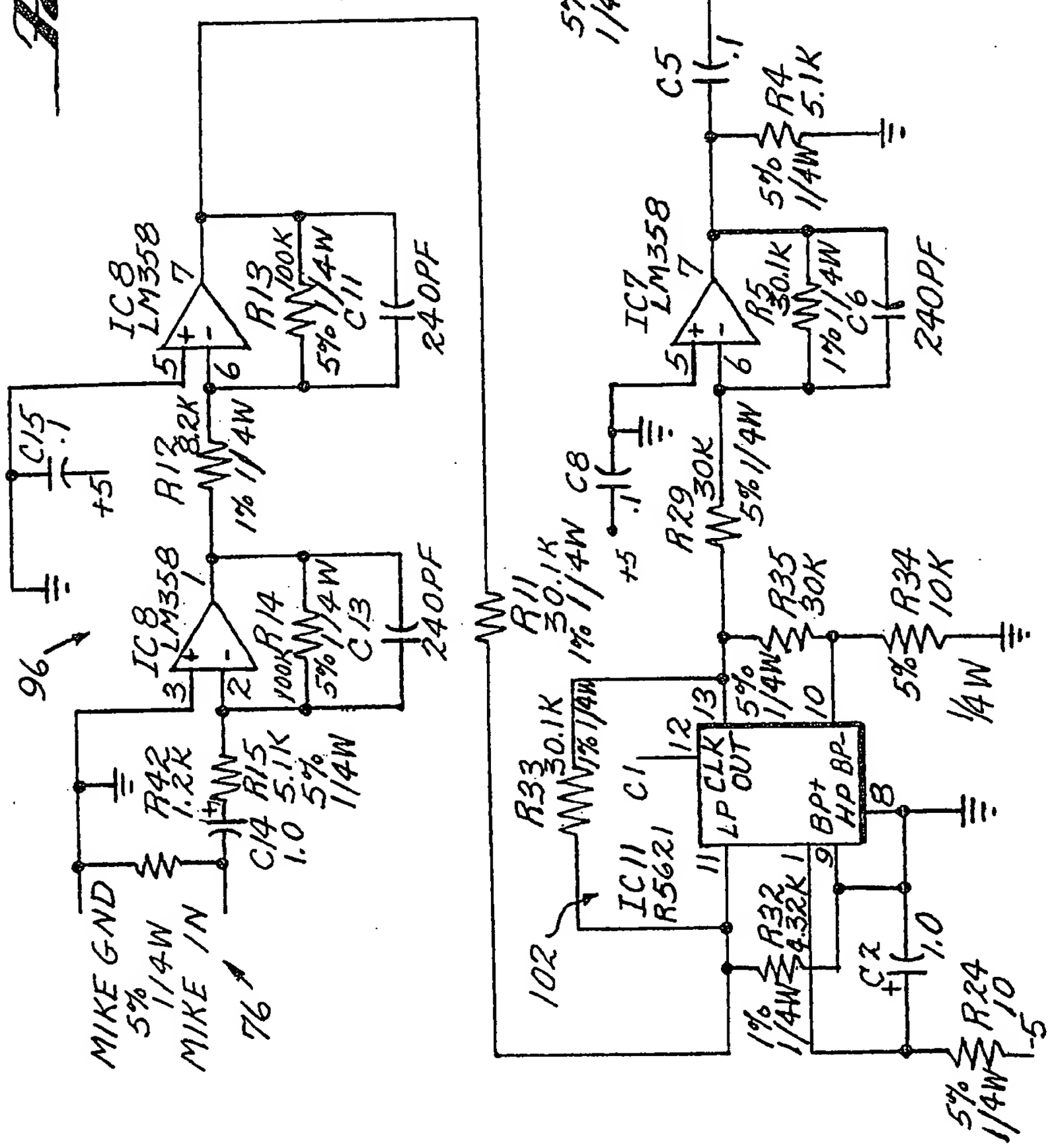
Fig. 4A.



Ridout & Maybee
PATENT AGENTS

1314395
19/4

Fig. 4B.



Kidout & Maybee
PATENT AGENTS

Fig. 5A.

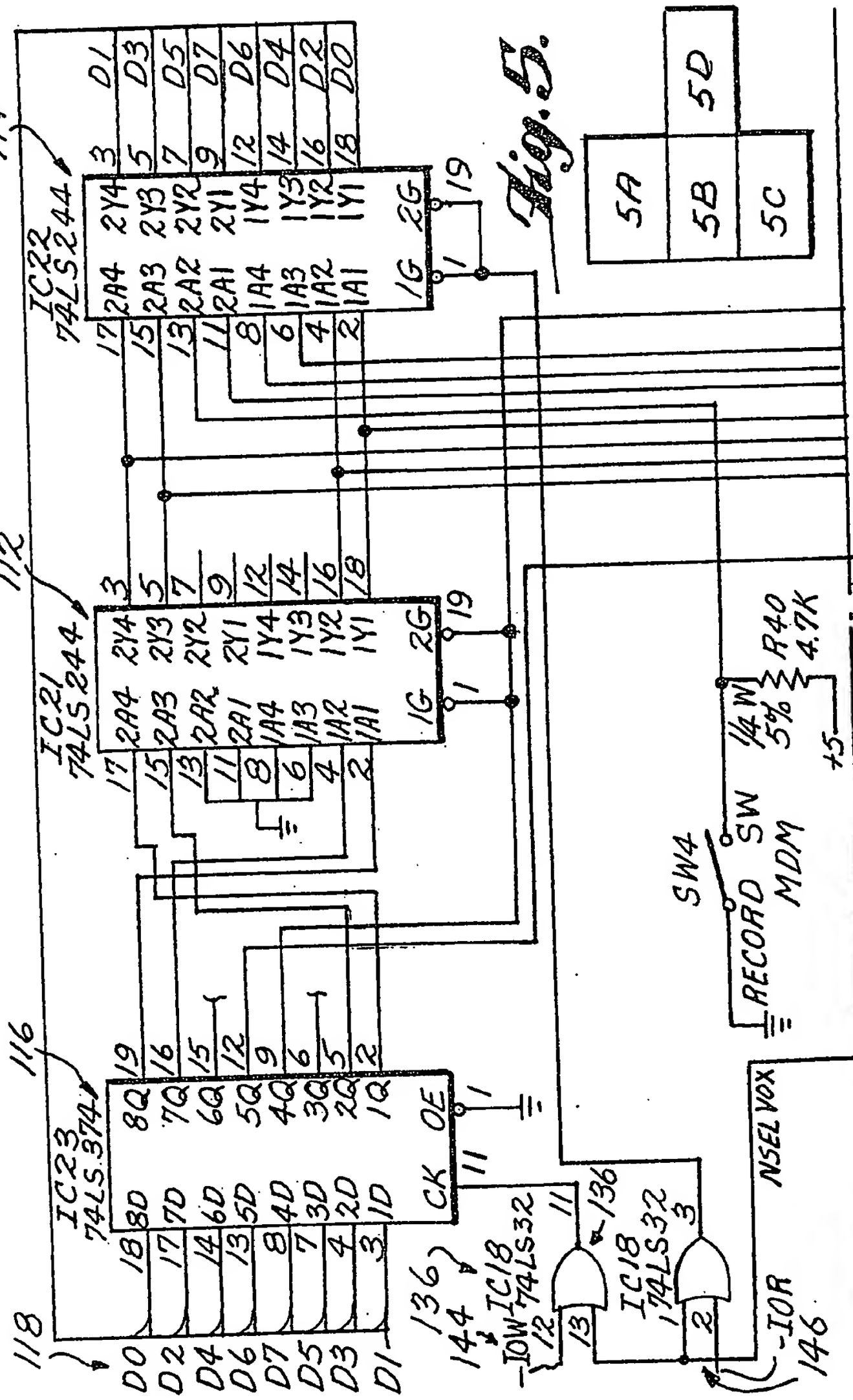
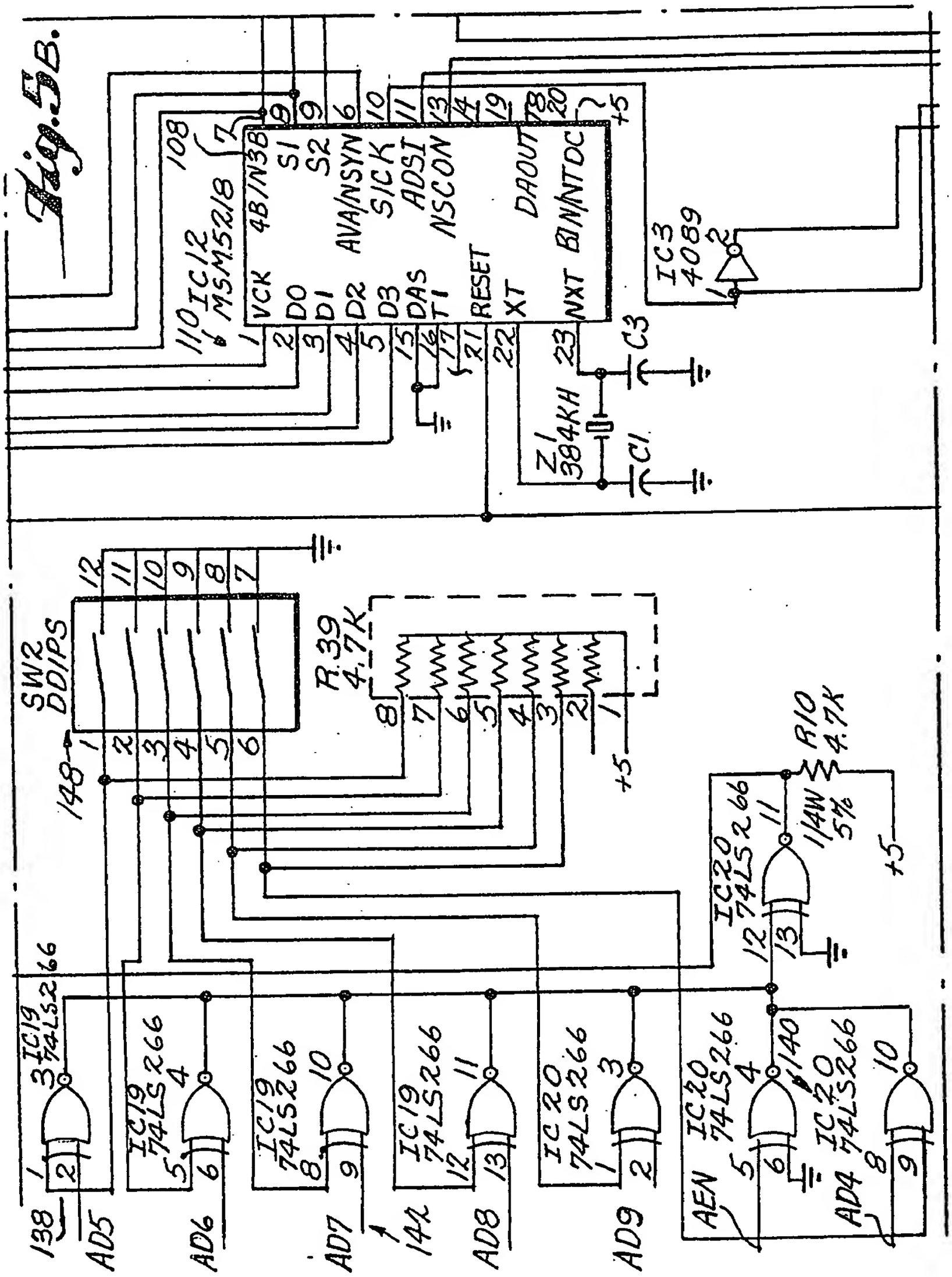


Fig. 5.

Ridout & Maybee
PATENT AGENTS

Fig. 5B.



Eidous & Mayhew
PATENT AGENTS

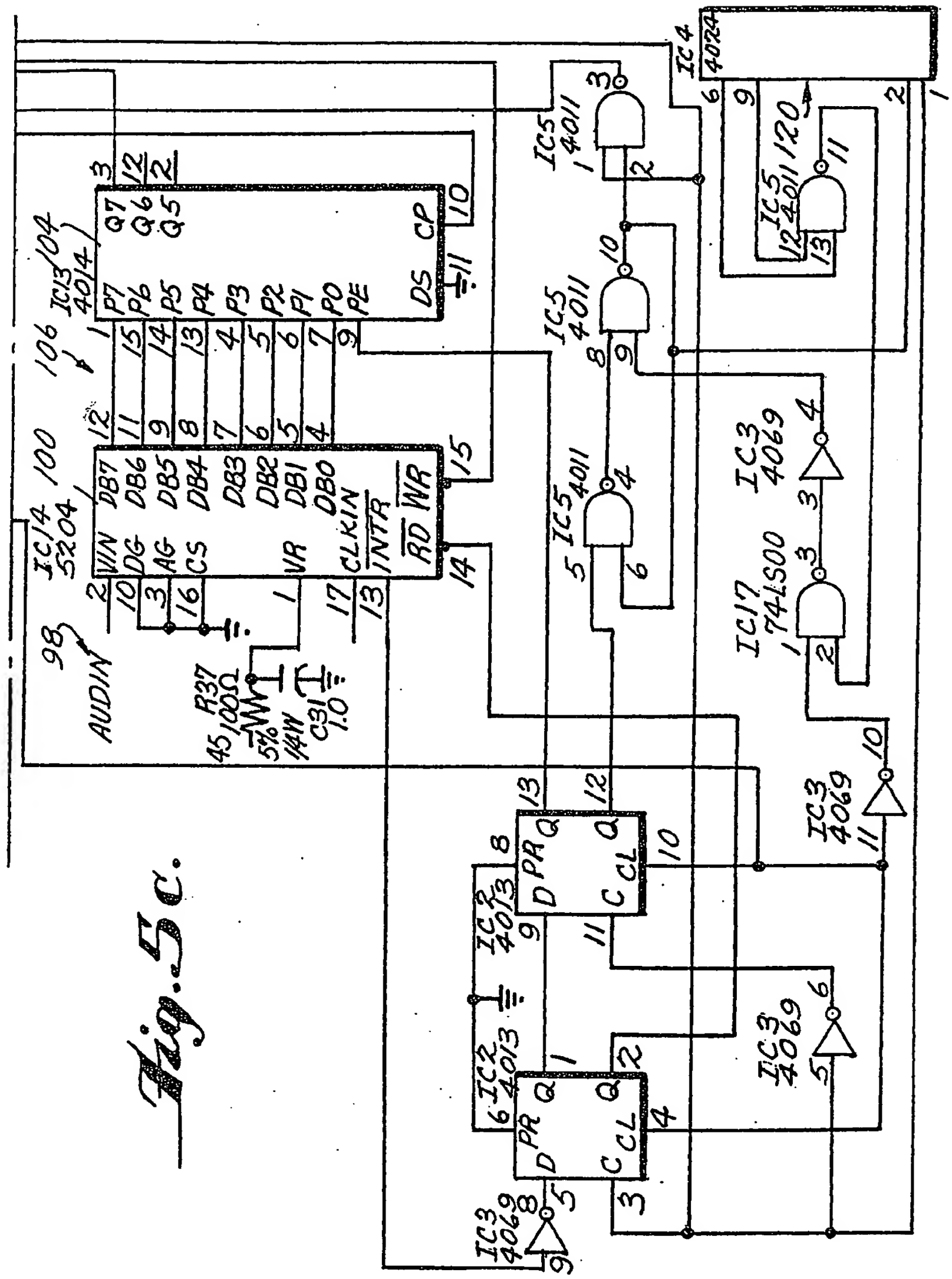


Fig. 5c.

Ridout & Maybee
PATENT AGENTS

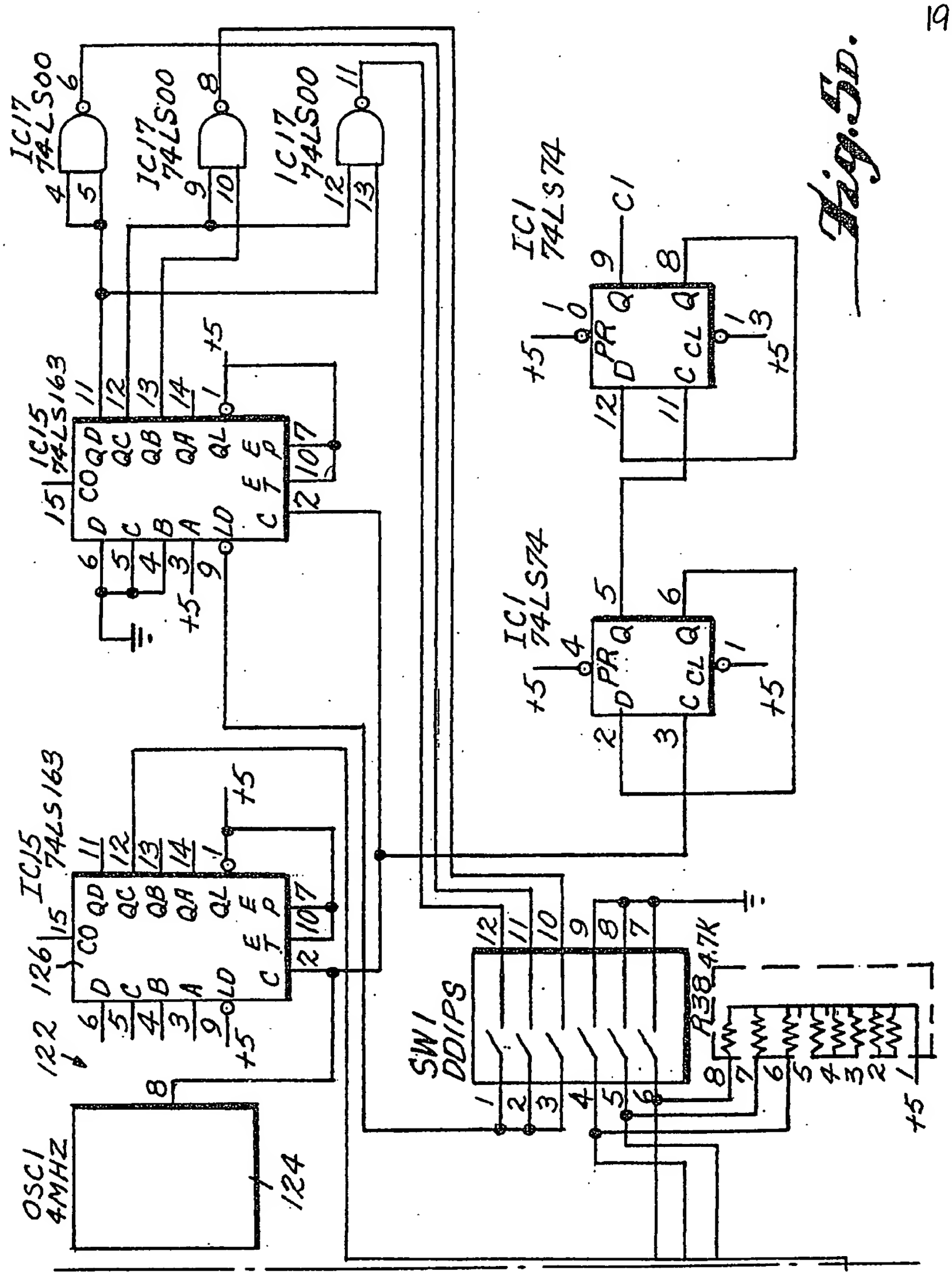


Fig. 5D.

Ridout & Maybee
PATENT AGENTS

Fig. 6.

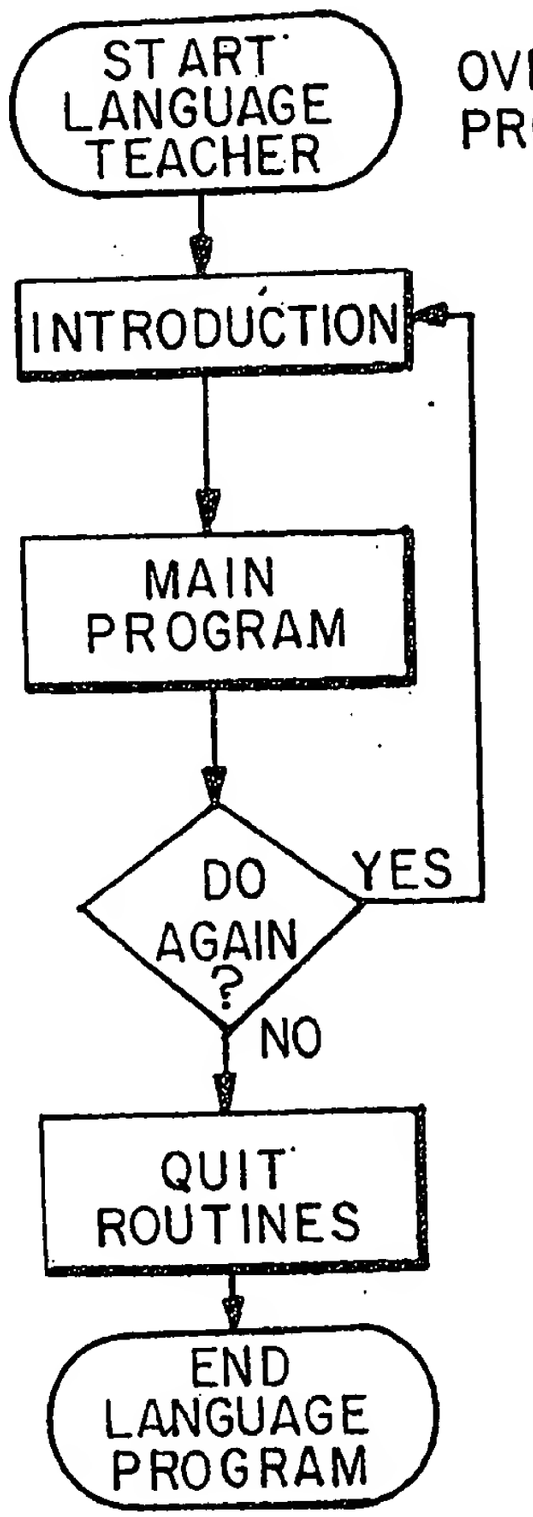
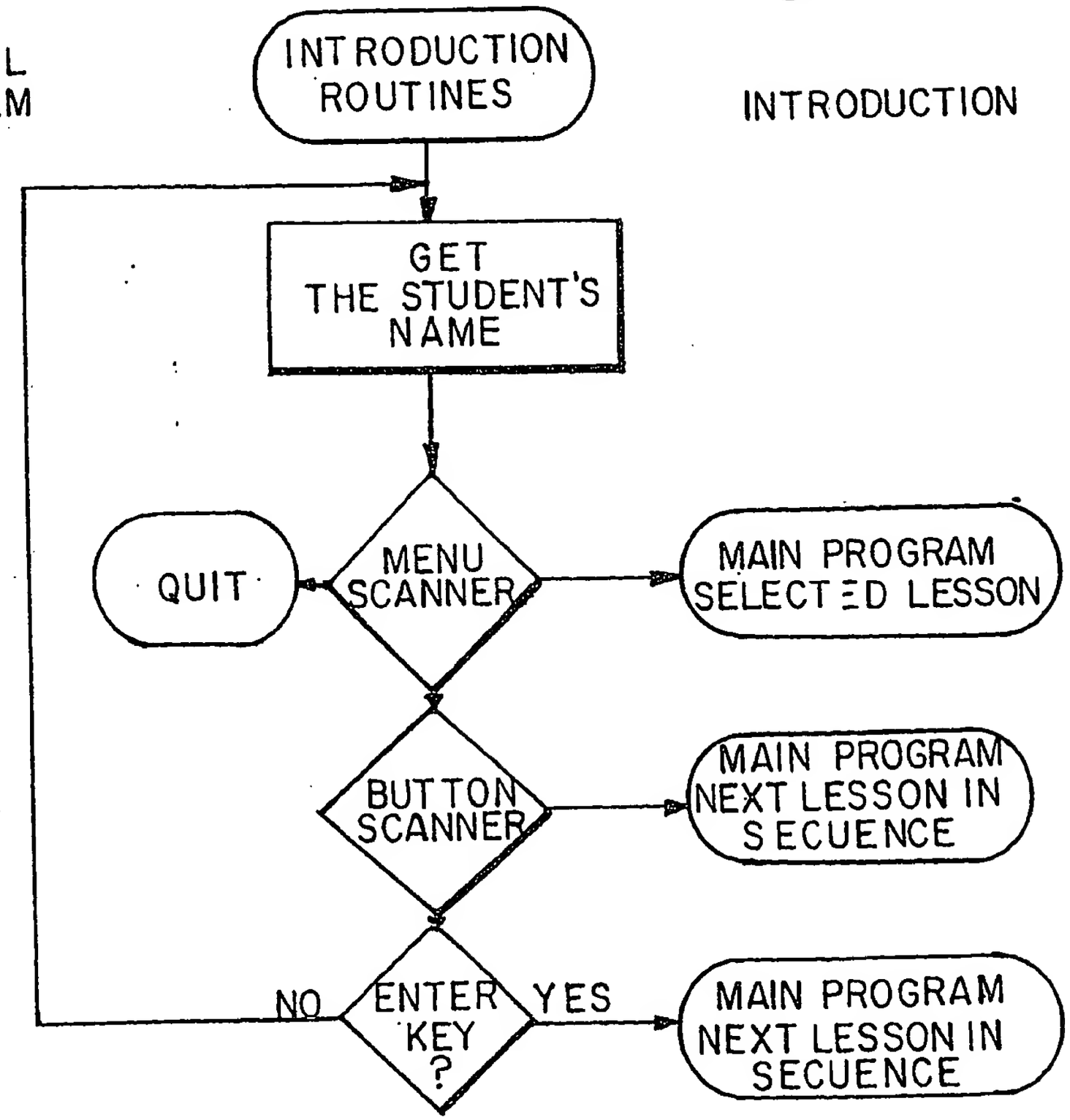


Fig. 7.



Rivise & Rivise
PATENT AGENTS

1314395
1919

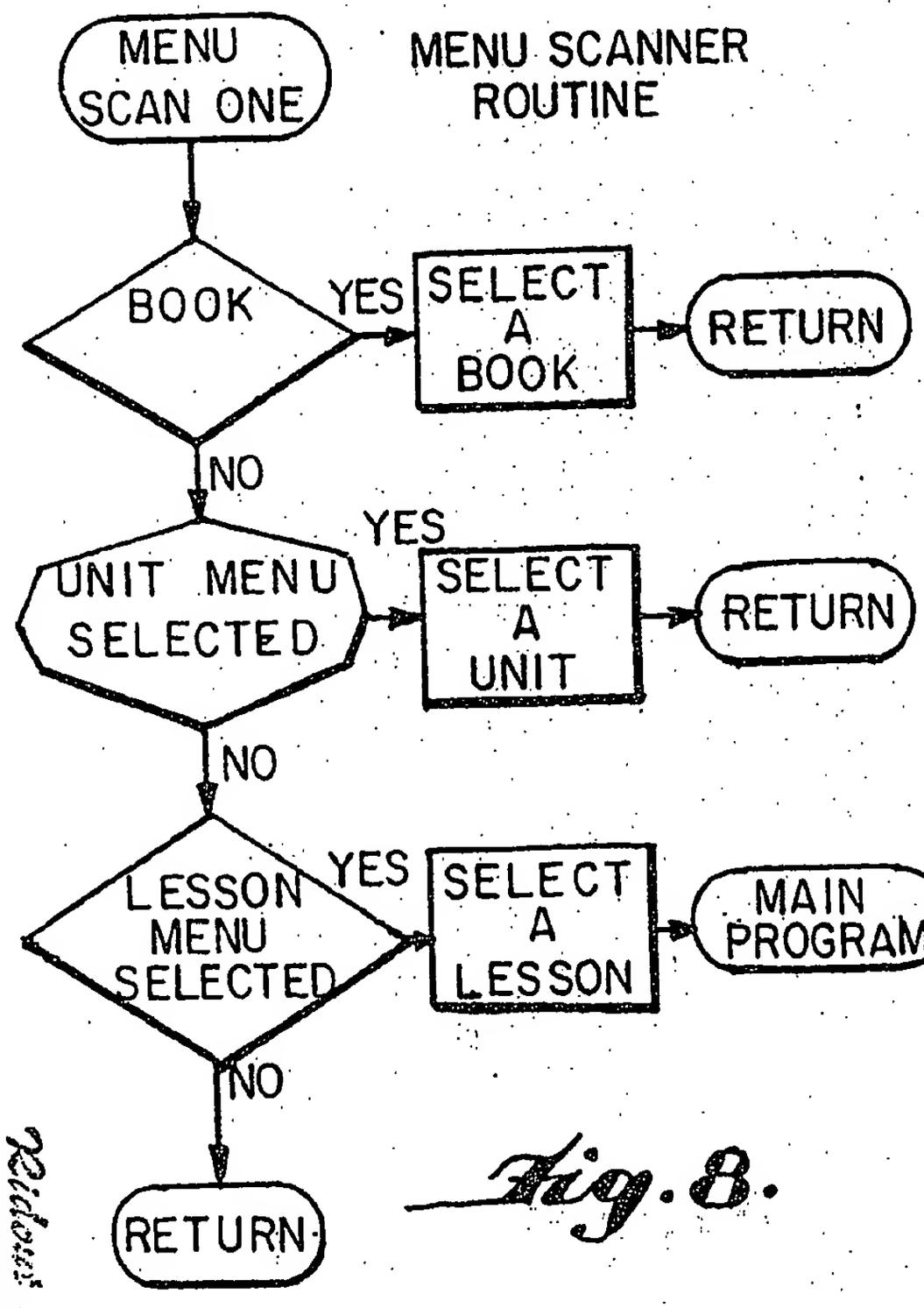
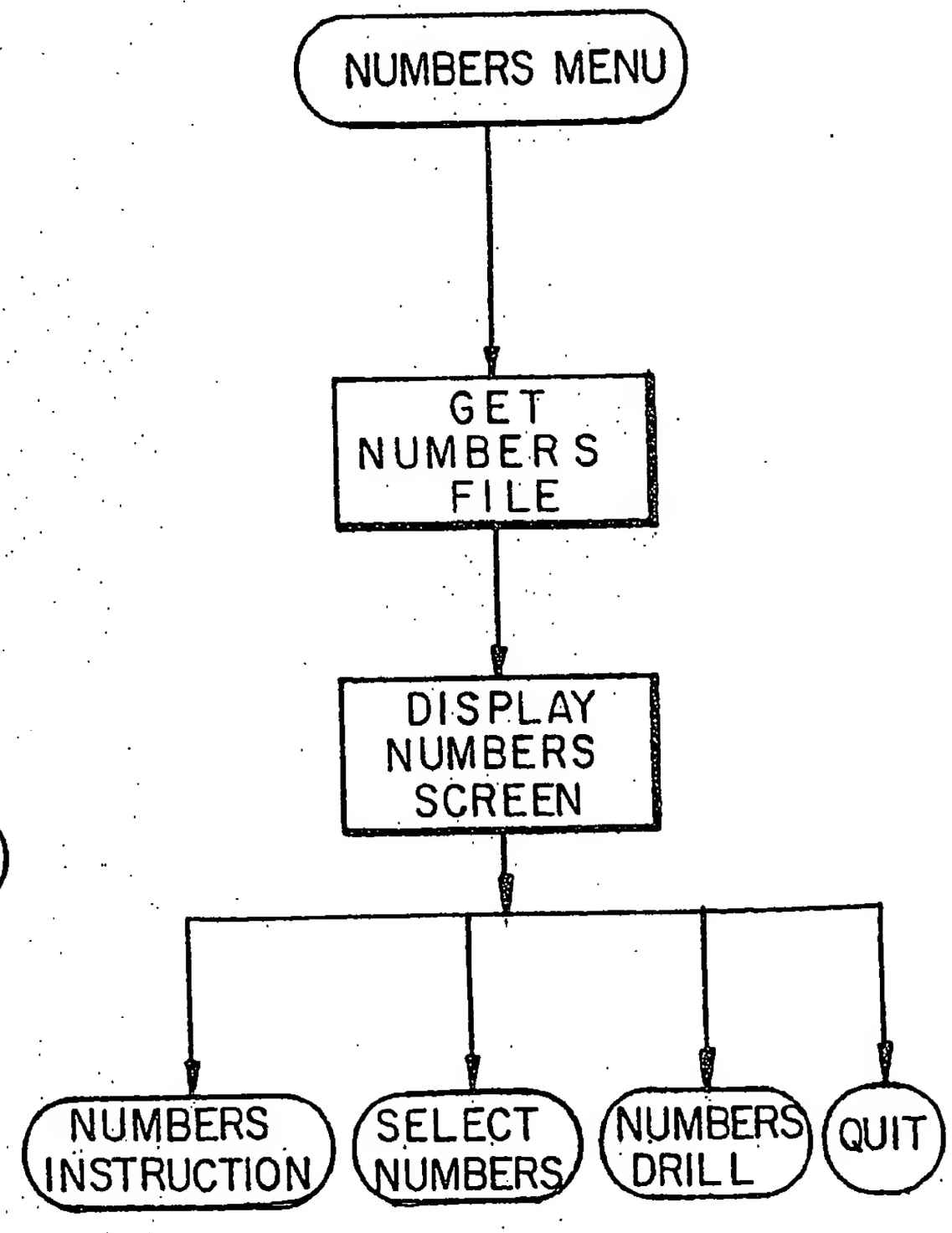


Fig. 8.

Fig. 9. MAIN NUMBERS PROGRAM

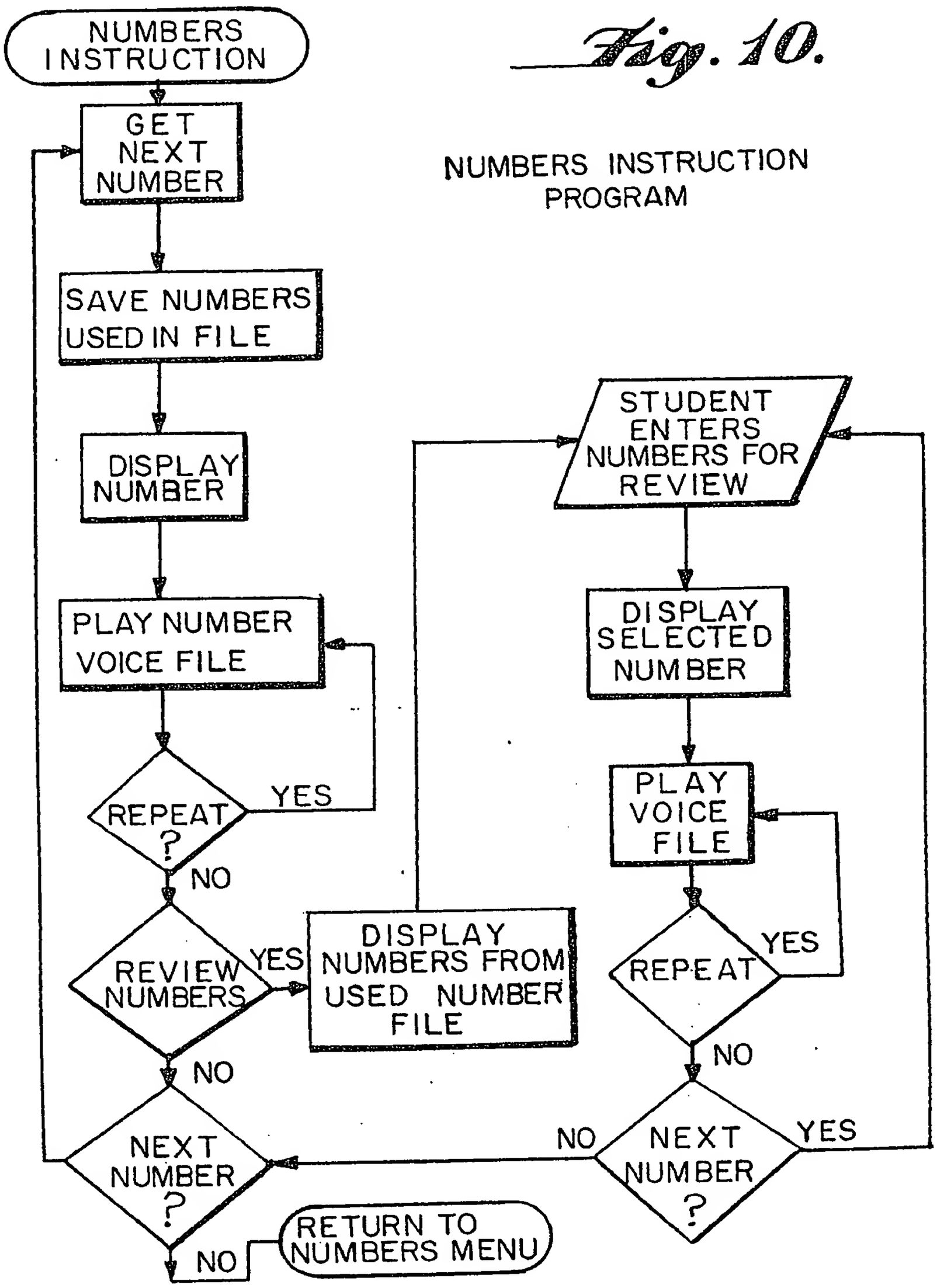


Robert G. Hodges
PATENT AGENTS

1314395
19/10

Fig. 10.

NUMBERS INSTRUCTION
PROGRAM



Kidons & Waybee
PATENT AGENTS

1314395
19|12

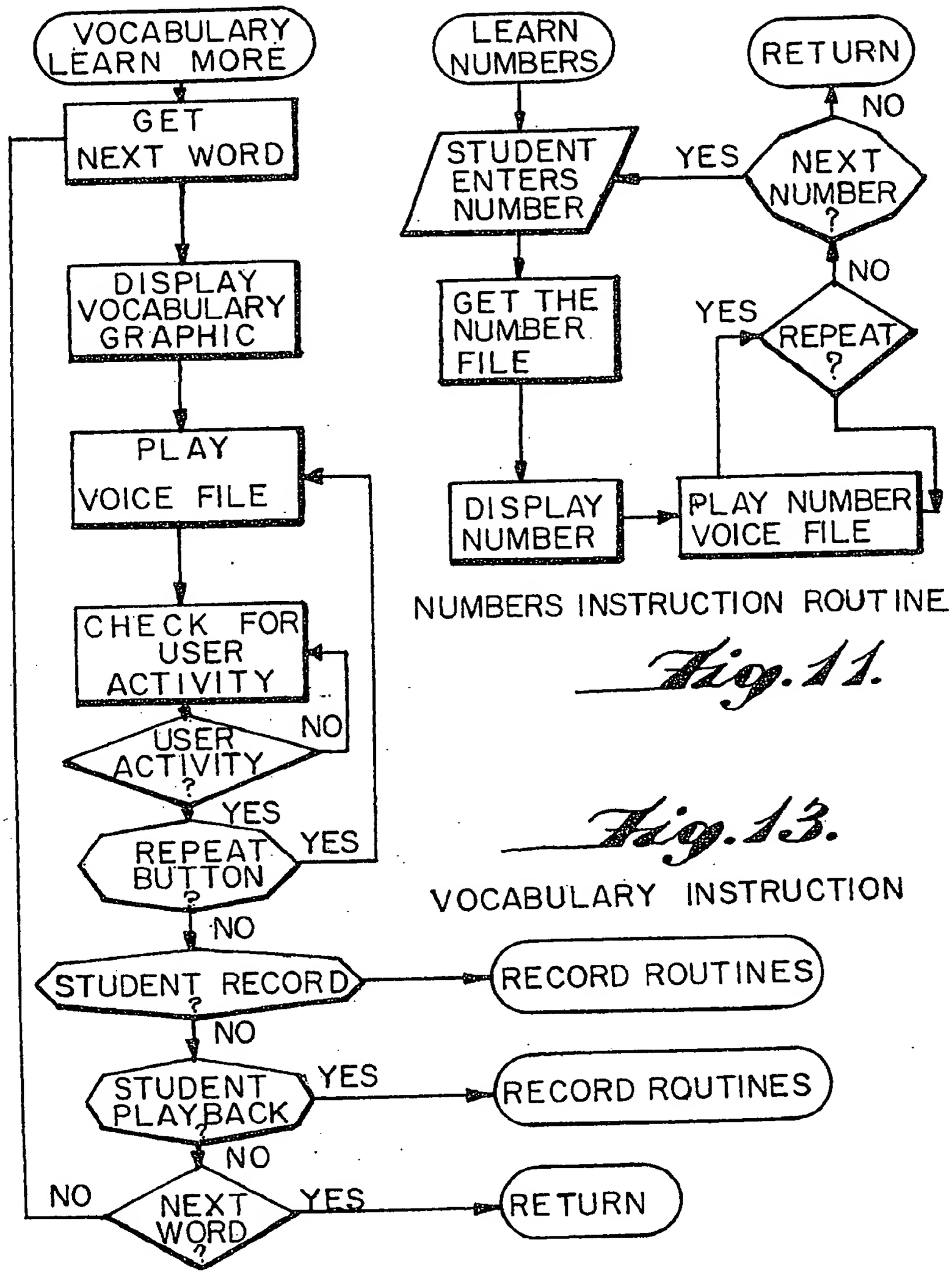


Fig. 13.

VOCABULARY INSTRUCTION

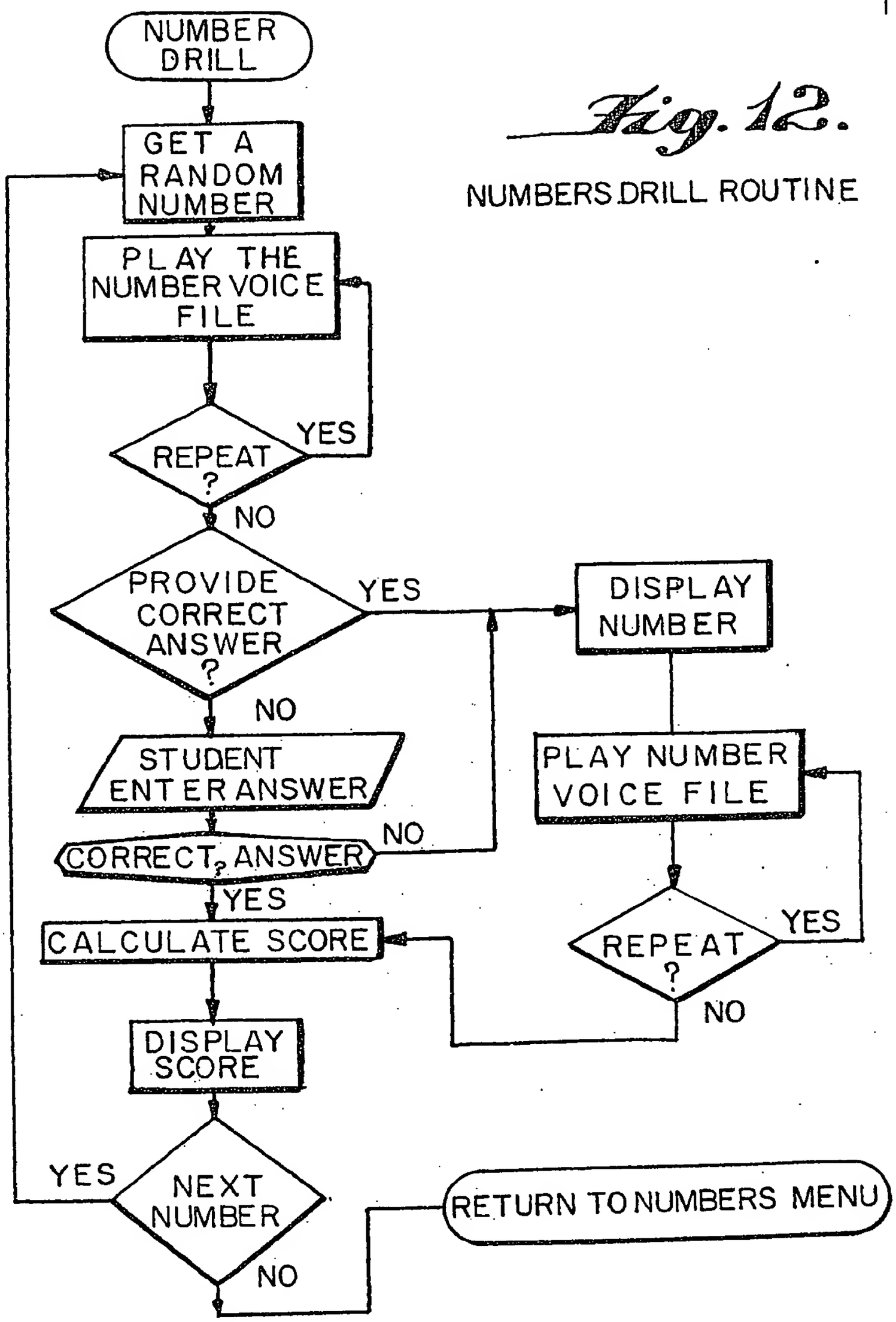
NUMBERS INSTRUCTION ROUTINE

Fig. 11.

Widow & Maybee
PATENT AGENTS

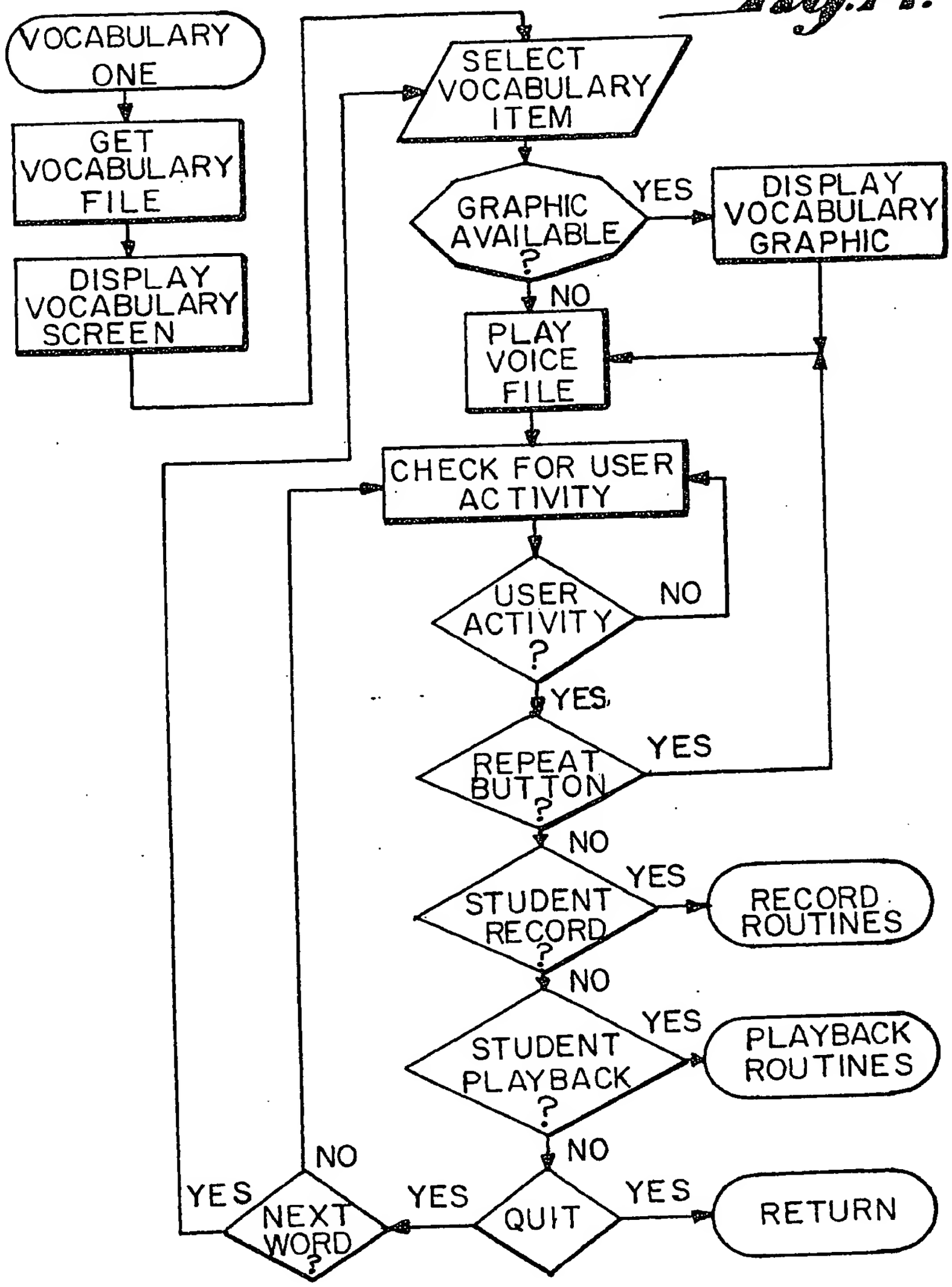
Fig. 12.

NUMBERS DRILL ROUTINE



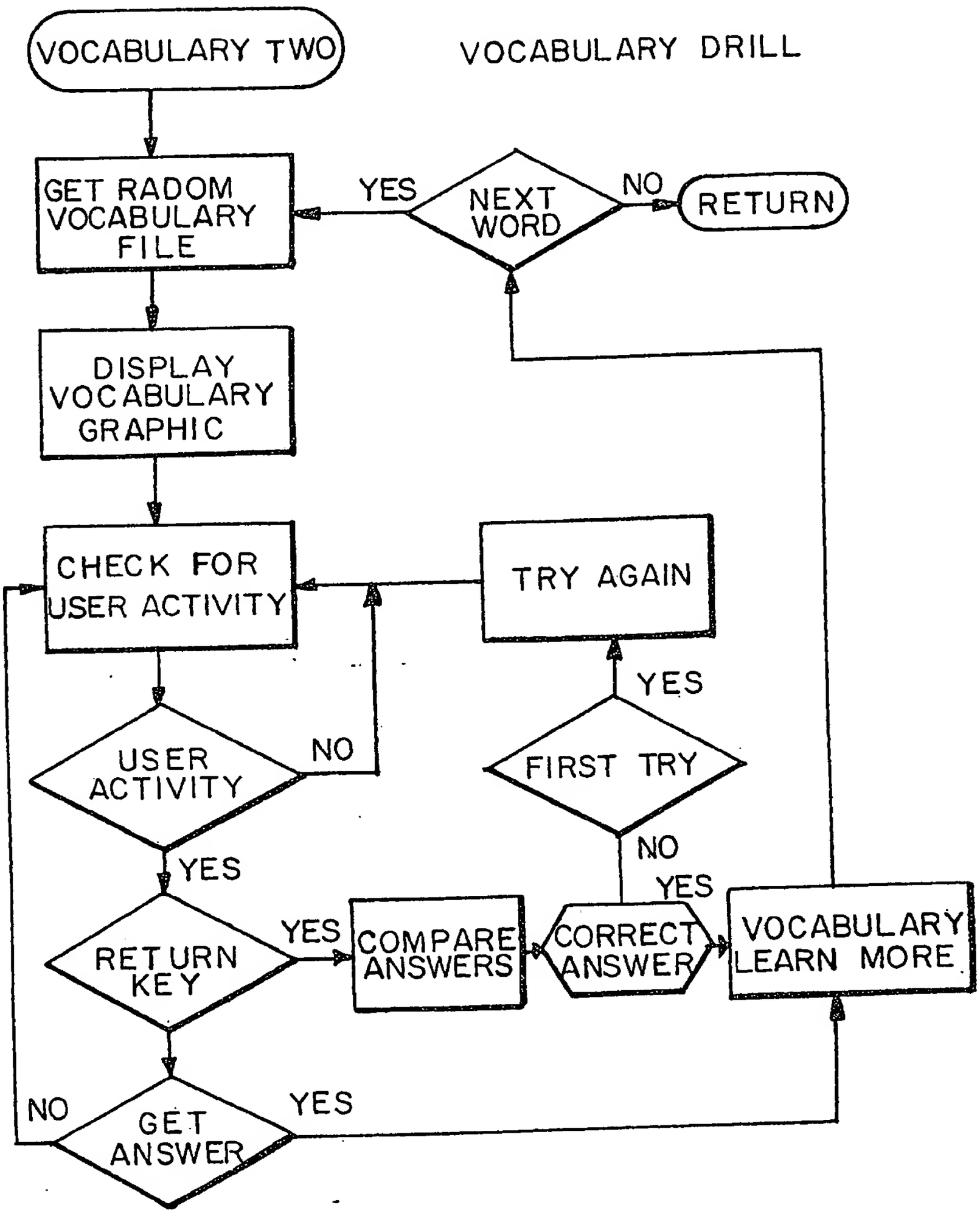
Ridout & Maybee
PATENT AGENTS

Fig. 14.



Ridout & Maybee
PATENT AGENTS

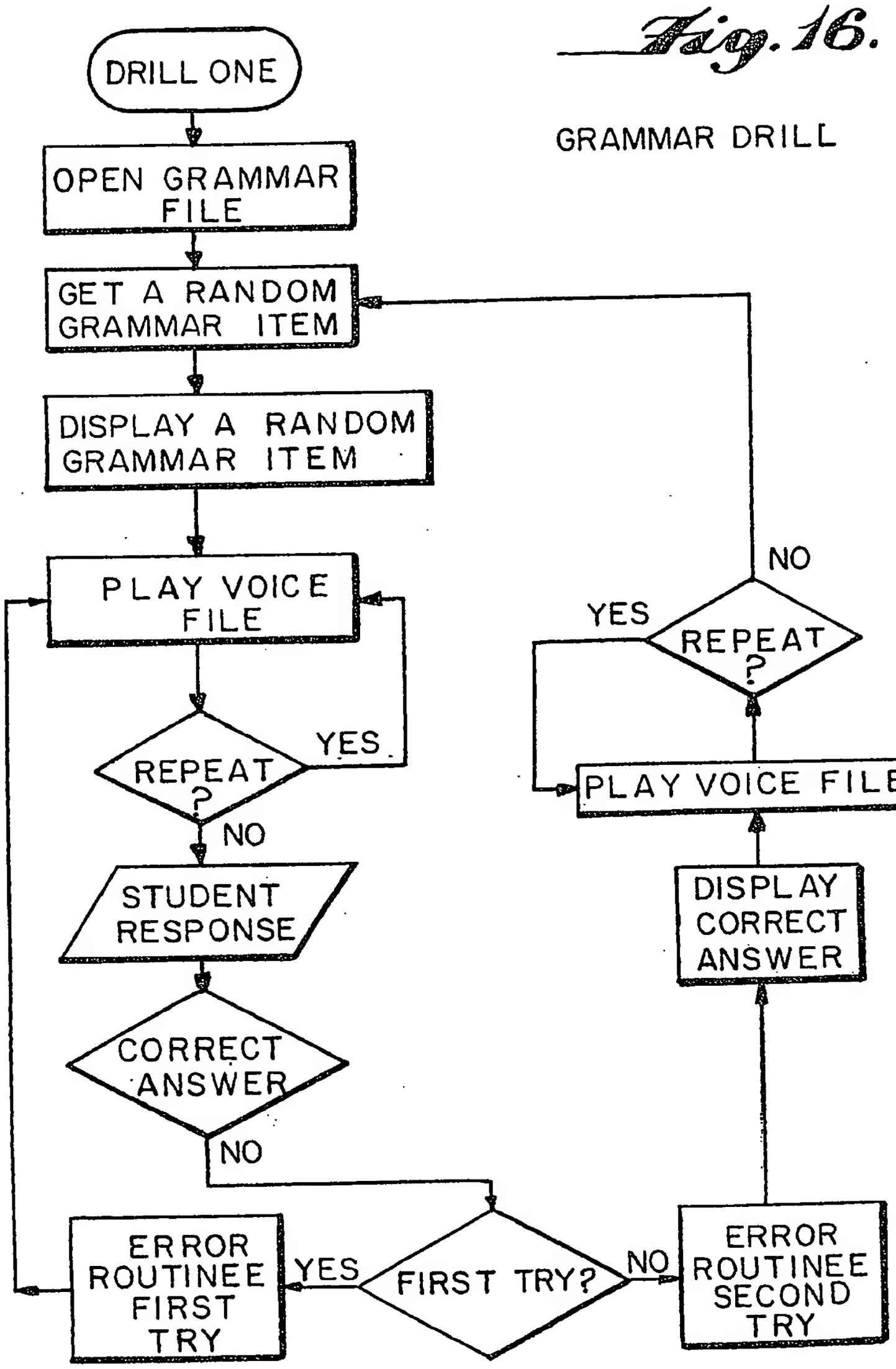
Fig. 15.



Kidon & Maybee
PATENT AGENTS

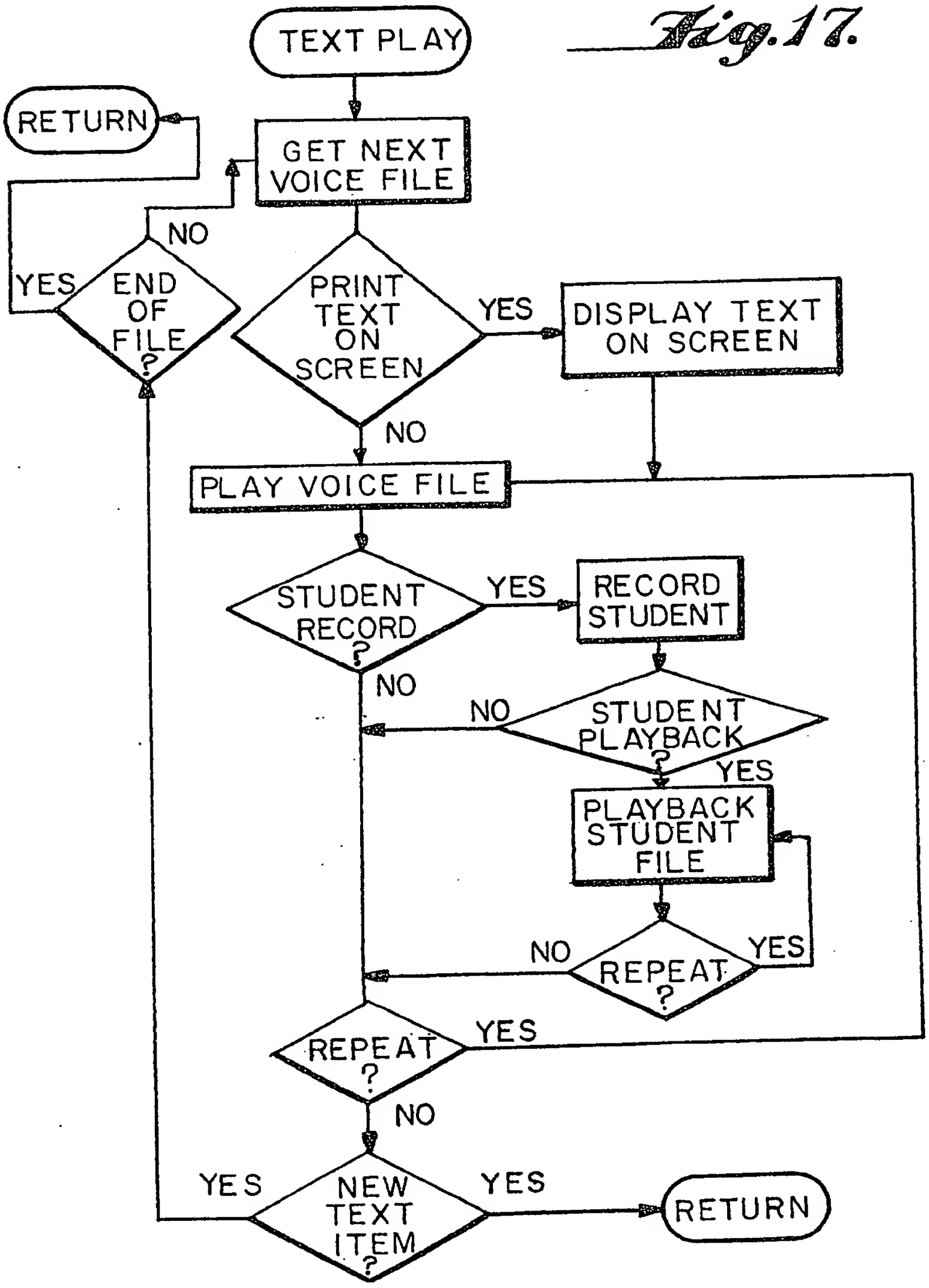
Fig. 16.

GRAMMAR DRILL



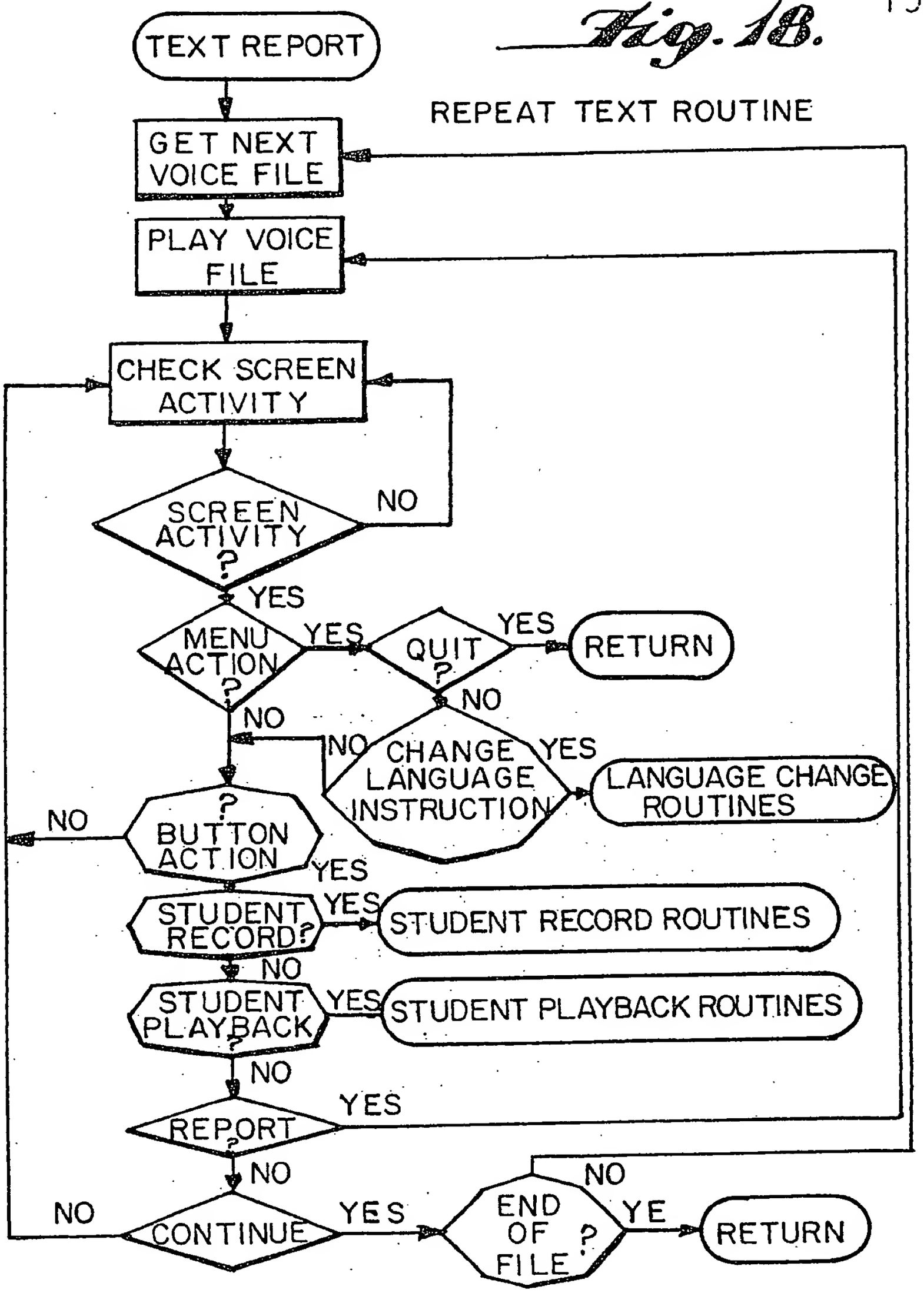
Ridout & Maybee
PATENT AGENTS

Fig. 17.



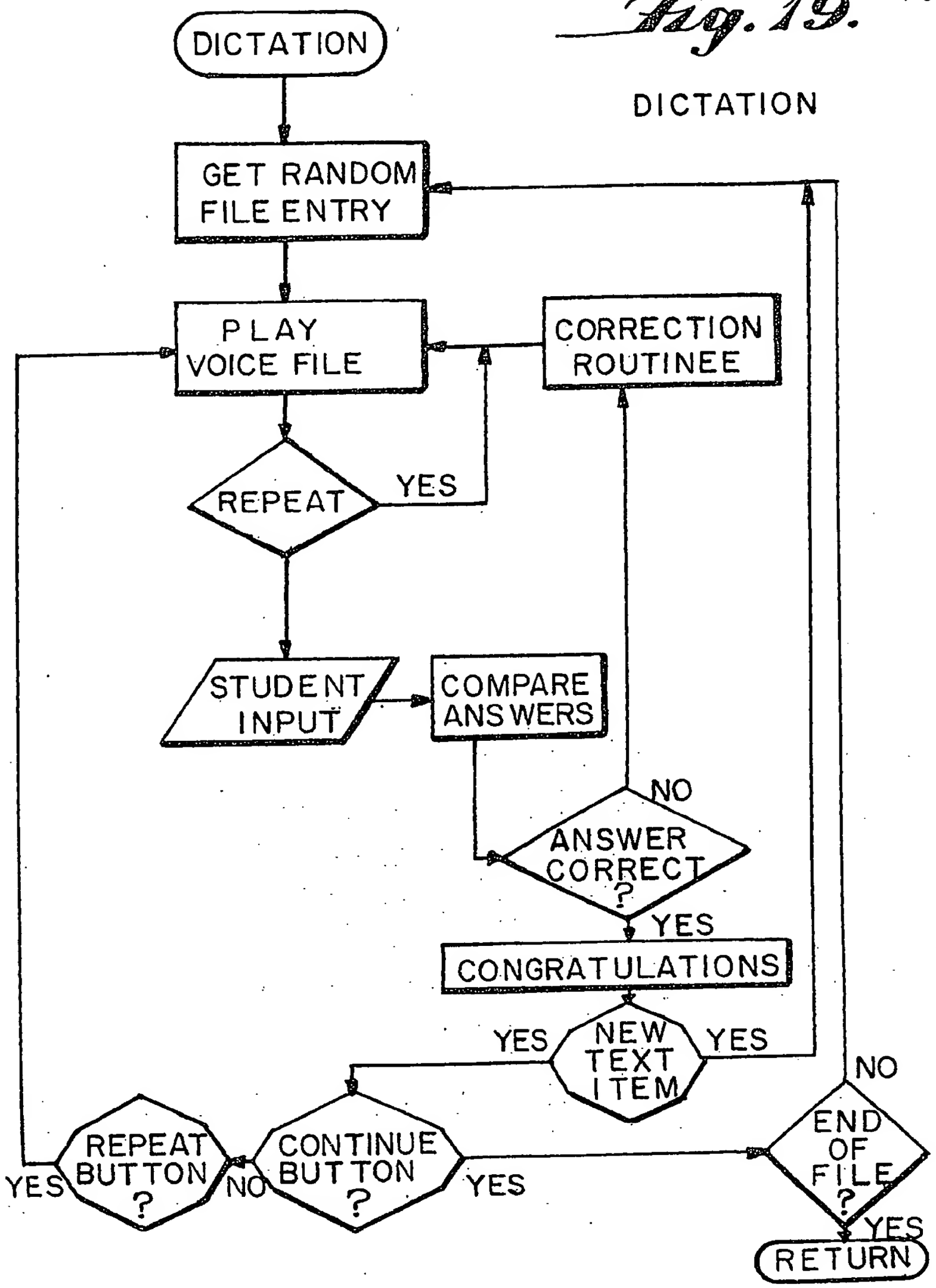
Ridout & Maybee
PATENT AGENTS

Fig. 18.



Ridout & Maybee
PATENT AGENTS

Fig. 19.



Ridout & Maybee
PATENT AGENTS